Graduate School Form 30
Updated 1/15/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Nikhil Rajan Tiwari

Entitled

Integrated Wireless Sensor System for Efficient Pre-Fall Detection

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Dr. Maher E. Rizkalla
Chair

Dr. Mohamed El-Sharkawy
Co-chair

Dr. Lauren Christopher
Co-chair

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Maher E. Rizkalla

Approved by: Dr. Brian King 4/10/2015

Head of the Departmental Graduate Program Date

INTEGRATED WIRELESS SENSOR SYSTEM FOR EFFICIENT PRE-FALL DETECTION

A Thesis
Submitted to the Faculty
of
Purdue University
by
Nikhil Tiwari

In Partial Fulfillment of the
Requirements for the Degree
of
Master of Science in Electrical and Computer Engineering

May 2015
Purdue University
Indianapolis, Indiana

To My Mom and Dad

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

ABSTRACT

Tiwari, Nikhil M.S.E.C.E., Purdue University, May 2015. Integrated Wireless Sensor System for Efficient Pre-Fall Detection. Major Professor: Maher E. Rizkalla.

The life expectancy of humans in todays era have increased to a very large extent due to the advancement of medical science and technology. The research in medical science has largely been focused towards developing methods and medicines to cure a patient after a diagnosis of an ailment. It is crucial to maintain the quality of life and health of the patient. It is of most importance to provide a healthy life to the elderly as this particular demographic is the most severely affected by health issues, which make them vulnerable to accidents, thus lowering their independence and quality of life. Due to the old age, most of the people become weak and inefficient in carrying their weight, this increases the probability of falling when moving around. This research of iterative nature focuses on developing a device which works as a preventive measure to reduce the damage due to a fall.

The research critically evaluates the best approach for the design of the Pre-Fall detection system. In this work, we develop two wearable Pre-Fall detection system with reduced hardware and practical design. One which provides the capability of logging the data on an SD card in CSV format so that the data can be analyzed, and second, capability to connect to the Internet through Wifi. In this work, data from multiple accelerometers attached at different locations of the body are analyzed in Matlab to find the optimum number of sensors and the best suitable position on the body that gives the optimum result.

In this work, a strict set of considerations are followed to develop a flexible, practical and robust prototype which can be augmented with different sensors without changing the fundamental design in order to further advance the research. The perfor-

mance of the system to distinguish between fall and non-fall is improved by selecting and developing the most suitable way of calculating the body orientation. The different ways of calculating the orientation of the body are scrutinized and realized to compare the performance using the hardware. To reduce the number of false positives, the system considers the magnitude and the orientation to make a decision.

1. INTRODUCTION

Fall in elder population of the age 65 and above has become a very frequent occurrence which has complex ramifications such as disabilities, loss of self-confidence, insecurities, financial losses and even death. In this advanced medical era where the life span of individual are long, there is a dire need to maintain the quality of life of the elderly. As the population keeps getting older the possibility of fall keeps increasing as the body gets fragile and the muscles start to weaken. The processes of aging is inevitable so there is a need to find a preventive measure to reduce falls. The work in ref [1] studied 1158 subjects for the risk of fall in people of the age 71 and above, who were staying in the community with demographic and medical characteristics factors of health care, cognitive, functional, psychological, and social functioning considerations. The study took the record of the number of days before the subjects were admitted into a nursing facility for a long period of time in the span of three years. The subjects were grouped into four categories: subjects with no fall, subjects who had one fall without serious injury, subjects with two or more falls without serious injuries, and subjects with at least one fall and a serious injury. A total of 133 people that represents 12.1 percent were admitted to a nursing facility for a long duration. The study showed that the risk of admission to a nursing facility increased at large for people who fell compared to people without any falls. The result of the study concluded that falls are an important factor of admittance in a nursing facility and systems that prevent falls may therefore delay or reduce the chances of nursing home admissions.

1.1 Advantages of Fall Detection

There are several researches in progress for detecting falls after they have happened which would only inform about the fall but will not prevent it. Even to monitor an elderly person there is a need of personal attention and proper data collection in order to continue further development and analysis which is mostly done manually by a caretaker, which can be flawed due to human error. The other difficulty is to record and monitor individuals continuously at different places which is very hard to do. If the fall has occurred, the only measure which can be taken is to rush the emergency services to the particular address, in which case the person can be provided with medical aide soon, but rather there can be a system which can detect a fall before it occurs, and this may act as a preventive measure. It would take time to come up with a complete solution which is perfect, but as progress is made in iterations, there can be a safety device system which can be efficient for commercial use. There is also a need of proper information about the falls, thus this system can record real time data about the ambulatory activities $24 * 7$ at remote location which could be of great help in building advanced Pre-Fall detection systems.

1.2 Literature Survey of Fall Detection

There is a large amount of research done and is in work with different fundamentals to develop a perfect system which would either detect a fall before it happens or detect a fall after it has occurred. There are multiple number of papers that are working on different ways to detect falls, some of them deal with the detection of falls after they occur and some on detecting the fall during the transition between balanced posture and the fallen posture that is during the fall. The other approach is to detect the fall before it happens. In order to detect the fall during its transition there is a need to understand the details of the pattern of falls. The first step in detecting a fall is by sensing the changes of the body position. There are multiple concepts applied to differentiate falls and normal activities using different electronic systems.

A Portable Pre-impact Fall Detector uses Inertial Sensors to detect a Fall before it occurs. The system uses tri-axial accelerometer and tri-axial angular rate sensor. For the data logger and for running the algorithm, the system here uses a Pocket PC(HP iPAQ h5550), the system can log 20 hours of data from the 3DM-G nine channel of 12 bit data stream at 57 Hz from each channel. In order to connect the Pocket PC to the sensors, LabVIEW 7.0 was used. The whole system is encased in a waist bag [2].

There were some research groups who tried to work with machine learning in order to predict the fall before it had occurred but was too complex to build for a portable embedded system with minimum hardware requirements [3]. Shan *et al.* [3] used feature selection and support vector machine to detect the fall earlier. A tri-axial accelerometer was used to measure the movement of the body and algorithms were used to select a proper feature which would give the maximum amount of difference between Activity of Daily Living (ADLs) and falls. The system used a STMicroelectonics LIS3LV02QD accelerometer to measure the acceleration of the body and a NEC 78K0547 Micro-controller as the brain of the system which executes the algorithm to detect the fall. The Micro-controller communicated with a PC which logged the data sent to it, using a wireless transmitter connected to the system. To transmit the data, a pair of Nordic Semiconductor nRF2401 wireless modules were used.

This research [4] deals with validating and developing an algorithm which uses 2D- information. The 2 Dimensions here are the trunk angular velocity(α) and trunk angle(ω). This work simulated unexpected slip-induced falls to validate and build the algorithm. The system uses an Inertia-Link Inertial Measurement Unit(IMU) which is put close to the sternum to measure the orientation, acceleration, and the angular velocity in 3-D, sampled at 100 Hz. Another sensing system which uses six-camera (ProReflex MCU 240) infrared motion capture to sense the position in 3-D, of the reflective marker and is also sampled at 100 Hz. To identify the falling motion, the position of the vertical marker was measured. Analysis was performed on the data set which was collected from the experiments to determine the discriminant function $F(\alpha, \omega)$ for the algorithm. The threshold is not associated with the physical sense

but as a composite score of integrated values. To determine the optimal value of the threshold for the algorithm 1, the ratio of sensitivity to specificity is used.

Some researchers have been working on the concept of using Smart-phone for detecting falls [5], [6], [7], some of which were specifically using android mobile operating system due to its openness. Yi Hi *et al.* [5] made use of the built-in accelerometer that collected the movement of the body to classify it into five categories i.e. vertical activity, lying, sitting or standing without movement, horizontal movement, and fall. The system used the other facilities provided by the device like sending Multimedia Messages (MMS) when a fall is detected which includes other information about the location, and the time at which it occurs.

Fang *et al.* [6] used the android platform to built a fall detection algorithm on the smart-phone alerting few selected contacts in an event of a fall. This study also analyzed the trade-offs between sensitivity and specificity, and provided information about the power consumption of the device. Tiwari *et al.* [8] used the android platform but let the user reduce any falls positives by touching the screen or responding in case of any falls positives. Sposaro *et al.* [7] worked on the same android platform and provided additional feature of sending a SMS to a pr-specified contact when a fall is detected and when the user gets a response back from the contact when the application puts the phone on speaker and let the user conforms the fall only, then the emergence services summoned.

Cheng *et al.* [9] looked into the Surface Electromyography and Acceleration in order to get information for classifying the activities, and detected the falls. To distinguish between different intervals of activity, the system used Histogram Negative Entropy, and in order to determine the posture of the subject, acceleration vector is used. This also helps in defining dynamic gait activities and dynamic transitions. The hidden Markov model was employed to identify the dynamic gait activities while the acceleration amplitude was used to detect falls. There are three major category of systems differentiated on the basis of the sensing mechanisms, those are Computer Vision based [10], Acoustic and Ambient sensor based, and wearable sensor method.

Each one of the methods have it's drawbacks and benefits [11], [12], [13], [14]. The approach with Computer Vision based systems is versatile but costly and confined to an indoor facility where the system was installed [15], [16]. In comparison, the approach using acoustic provides cost effectiveness but has its own drawbacks like inefficiencies due to external noise [17], [18].

Most of the previous research concentrated on detecting falls and not on pre-falls. One of the important concerns when developing any embedded system is the amount of hardware, which should be minimal in order to reduce cost, size and weight as it is to be carried by the subject. Narasimhhan [19] used a skin-contact sensor which consisted of a tri-axial accelerometer, a micro-controller and a low energy Blue-tooth transceiver. The algorithm took into account two variables, the magnitude variation, to tell if an impact has occurred and angle to check if the direction is horizontal, and confirm that the fall has occurred. The activity is then measured using a threshold to designate the fall. The sensor used here was a Bosch BMA250 digital tri-axial accelerometer. The accelerometer sampled the data at 125 Hz with a resolution of 10 bits and a range of $\pm 4g$ for each axis. The system worked with a microcontroller on the subject, which ran the algorithm and sent the signal when a fall was detected, using a low power Bluethooth transceiver.

Bevilacqua [20] used an approach on the lines of Vision based fall detection. Provided all the benefits and drawbacks of vision based approach. The research distinguished falls by evaluating the contraction and expansion speed of the volume of the object, using a concept of Human Bounding Box and also taking into account the position of the subject in space. The system worked on two inputs from a RGB-D camera and Microsoft Kinect device. The system used IR depth sensor and a RGB camera both of 640×480 resolution at 30 fps from PrimeSense which also provided the software library. As the system worked on the Microsoft Kinect IR sensor there was a limitation on the range of view and depth which comes to 3.5m specified by Microsoft. The range was also a factor of the environment and the illumination condition, thus providing inconsistent results.

Prado-Velesco *et al*. [21] put forward a strategy called divide and conquer to detect the occurrence of a fall. The system that was developed for fall detection was unobtrusive and not user friendly. As the system had to be worn by the subject for long durations, it should be designed with a level of comfort. The understanding of the real world fall pattern was limited to some extent and very complex as there were unlimited number of factors to be considered in real world scenarios. The study suggested that this complexity cannot be addressed by using rigid analysis of acceleration. The system was based on an adaptive algorithm in order to differentiate between impact and non-impact activities. The study showed that there is a range of threshold which provides a balance between specificity and sensitivity. The study also showed that the threshold can be reached by tuning during normal daily activities. The system had two layered architecture, the intelligent accelerometer sticking plaster (IASP), and the wearable base station connected to a smart-phone. Both systems were connected by wireless personal area network using a free media access control (MAC). The complexity of this system, and the increased hardware was a huge drawback and would not bring fall detection to commercial market.

There are some researches focusing on developing different designs for the fall detection systems so that the system can be worn comfortably without any obstruction in daily living. This particular approach is very important as the system may be worn on for long durations in different scenarios, such as while taking bath, etc. The system should also be capable of handling impacts and be resistant to water as it will be worn on at all times. Very few researches have taken considerations of these facts. Bourke *et al*. [22] developed a vest consisting of the fall detection system. Their approach for the fall detection was threshold and posture recognition based, the two conditions that were checked before making the decision. The system used a Freescale MMA7261QT tri-axial accelerometer, MSP430 microprocessor a Micro SD card and a Bluetooth module. The accelerometer data is sampled at 100Hz and filtered using 1^{st} order analogue low-pass RC filter. This system was fixed onto the vest which would be worn on by subjects of different sizes. To solve the issue, the vest was made in

different sizes. Although the approach of using a vest to fix the system comfortably on the subject was promising, but the robustness of the system was still minimum as it could not be worn while in the shower.

Igual *et al*. [23] surveyed the fall detection research classifying the technique into two categories, context-aware systems which consisted of sensors implemented in the surrounding of the subject rather than on the user, giving a comfortable and free environment for the user, but limiting the mobility to a specific area such as a room or a clinical facility. The other drawback was the privacy, as these systems mostly used video based mechanisms to detect the changes of the posture. The 2^{nd} category was the wearable devices which included sensors worn by the subject. There were two sub-categories, one fell under the machine learning algorithms and second were threshold based. The machine learning approach provided good results but with increased complexity of hardware and software, whereas the threshold based approach gave average results, depending upon the execution of the design with benefits like simplicity of the design.

There are great opportunities in all of the above approaches with different drawbacks. The threshold based system with inherent simplicity provides a more viable option in order to develop commercially appealing systems which have the potential for mass usage. In all of the above approaches, none of them could fully accommodate the patients needs.

1.3 Feature of Wireless Devices

The importance of connecting systems to a centralized hub such as the Internet is undeniable as it gives a freedom to retrieve and manipulate the data from anywhere in the world. The Pre-Fall detection system can also be used to acquire data from subjects in real time and store it in a centralized place to be used for further analysis. Due to the advances in Internet speeds and WiFi technologies many of the systems can communicate with each other to leverage maximum benefit. This level of

communication between systems gives endless possibilities for their applications. The information about the surrounding from different connected devices gives great control of the environment which adds to the capacity of regulation. Another advantage of the Internet of Things is that the devices can be managed or controlled through applications built in the smart-phones. The data which is sent to website can be used by the smart-phone to inform individuals of the ongoing situations miles away, and may even give control over the situation by giving authority to induce changes.

1.4 Embedded Sensor Applications and Benefits

Any fall detection system requires information about the physical quantities such as temperature, position, acceleration, pressure, humidity, altitude, orientation etc. and the changes of the system relative to the surrounding needs to incorporate sensors in the system. The electronic devices can detect electrical signals only, but the physical quantities are not in this natural form, so they have to be converted in the form that can monitor these physical quantities, the sensors perform these transformation and provide the electronic machinery with the electrical equivalent. In all the applications in the home, automobile, aviation, medicine, manufacturing etc. devices need to be aware of the changes in the surrounding either to control, measure, or store the information that is acquired by the sensors. The need of sensors requires the chips to be versatile in every aspect from electrical to physical and mechanical stand point. The advancements in fabrication technology help achieve many required traits giving a low cost solution with superior precision and accuracy. The latest fabrication technique makes it easy to add more number of transistors on the same die which gives more real estate for adding complex logic. Apart from the benefit of size, the reduction of the transistor gate length reduces the power dissipation too, which is one of the basic requirements of any mobile device which runs either on battery or solar energy. There are a few common forms of processing like filtering, analog to digital conversion which are to be performed on the data provided by the sensors,

and normally is taken care of by the micro-controllers, but due to the optimum usage of chip real estate, these processes can be accomplished on-chip by adding computational engines that reduce the burden on the micro-controller. Another benefit is that we can add multiple sensors on one chip that may further reduce the complexity, power consumption, size, and cost of the device.

Few of the widely used sensors are Accelerometers and Gyroscopes which give the information about the movement to the system. Any hardware which requires this information, linear or angular, will need to include these sensors. These sensors are fabricated with different technologies and these varieties have there benefits and drawbacks, depending on the use and the type of systems which incorporate them. The selection of a proper type is very important for a perfect design as it affects the project's complexity and compatibility.

1.4.1 Types of Accelerometers

1. **Capacitive**: these types of accelerometers register acceleration due to the change in the distance between two plates of a capacitor.

2. **Piezoelectric**: These accelerometers work on the principle of Piezoelectric effect. There are Crystals which generate electricity on one side when pressure is applied on the other two faces of the crystal.

3. **Piezoresistive**: The working is similar to Piezoelectric accelerometer, but this strain gage is used, provide variation in acceleration due to change in resistivity.

4. **Hall Effect**: This works on the principle of change in the magnetic field due to change in acceleration.

5. **Magnetoresistive**: These accelerometers work similar to the Hall Effect accelerometers but measure the change in resistance due to a magnetic field which intern is represented as acceleration.

6. **Heat Transfer**: There is a single heat source around with thermo resistors was placed equally spaced so the temperature gradient is symmetrical until a change in acceleration occurs.

7. **MEMS-Based**: Micro-Electro Mechanical System Based accelerometers. These contain small Electro-Mechanical components which read the acceleration. This is the latest in sensor technology.

1.4.2 Types of Gyroscopes

Gyroscopes are sensors which measure the change in angular rate. There are many types of gyroscopes depending upon the basic technology used to develop them. Three basic types are:

1. **Spinning Mass**: This type of gyroscopes works on the principle of conservation of angular momentum. There is a spinning wheel within frictionless gimbals which resists the change in its angular momentum and measures the change in angular rate.

2. **Optical**: This type of gyroscopes uses optical principles such as Sagnac effects to measure the angular rate change.

   (a) Fiber Optical Gyros.

   (b) Ring Laser Gyros.

3. **Vibratory**: Most of all MEMS gyroscops are based on these principles.

   (a) Vibratory Coriolis Angular rate Sensor.

   (b) Basic Planer Vibratory Gyro.

1.5 Serial Communication

There are few constraints in the design of embedded systems as the purpose of these devices are to be mobile, low power, small in size, reliable, convenient etc. and due to this fact there are different technologies developed to fulfill these requirements. The Embedded system is a group of devices communicating with each other providing relevant information required for the completion of a task. The different devices are communicating in a serial manner bit by bit, either in a synchronized fashion or an asynchronous way, depending upon the protocol and the device. The reason to have a serial communication is to reduce the number of wires and ports on a processor or any communicating device. For these reasons two of the serial communication protocols have been extensively used due to their benefits;

1. Inter Integrated Communication (I2C)
2. Serial Peripheral Interface (SPI)
3. Universal Asynchronous Receiver Transmitter (UART)

2. HARDWARE

2.1 Arduino Board

Arduino is an open-source electronics hardware platform based on easy to use hardware and software. There are variety of hardware vendors developing Arduino Boards with different types of Micro-Controllers. The following are a few types of Micro-Controllers which are supported by the Arduino platform.

1. **ATMega328** clocked at 8 or 16 MHz
2. **ATMega1280** clocked at 16 MHz
3. **ATMega2560** clocked at 16 MHz
4. **ATMega32U4** clocked at 16MHz
5. **SAM3X**

2.2 Arduino Uno

Arduino UNO is a board built around the ATMega328 Micro-Controller. It has 14 I/O pins with configurable functions, and 6 of which can be used as Pulse Width Modulated (PWM) output, and 6 analog inputs. The ceramic resonator runs at 16 MHz, and it has a USB (Universal Serial Bus) connection to communicate with the PC, a power jack, an ICSP header, and a reset button. It contains everything needed to support the Micro-Controller, and it can be connected to a computer with a USB cable for programming or powering the board. It can also be powered by a AC-to-DC adapter or battery to get started. The UNO differs from all preceding boards, in that it does not use the FTDI USB-to-serial driver chip. Instead, it features

the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. The Version R3 is used in this Prototyping. Figure 2.1 shows the Arduino board with its components.

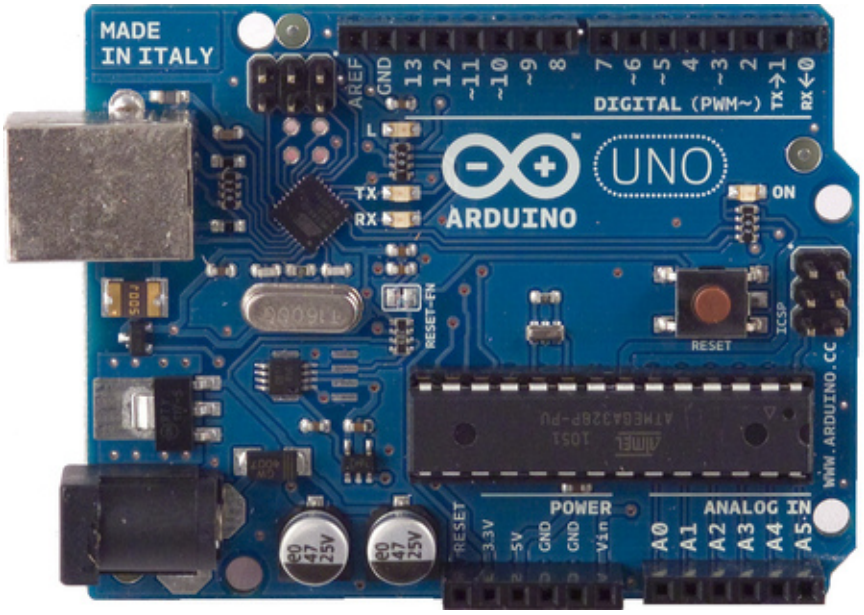

Figure 2.1.: Arduino UNO

2.3 ATMega328

The Atmel ATMega328 is an 8-bit AVR RISC-based microcontroller which combines a 32KB ISP flash memory and read-while-write capability, 1KB EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI (Serial Peripheral Interface) port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes.

Table 2.1.: Features of the Arduino UNO

Microcontroller	ATmega328.
Operating Voltage	5V.
Input Voltage (recommended)	7-12V.
Input Voltage (limits)	6-20V.
Digital I/O Pins	14 (of which 6 provide PWM output).
Analog Input Pins	6.
DC Current per I/O Pin	40 mA.
DC Current for 3.3V Pin	50 mA.
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader.
SRAM	2 KB (ATmega328).
EEPROM	1 KB (ATmega328).
Clock Speed	16 MHz.

The device operates between 1.8-5.5 volts. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed. Table 2.1 shows the features of the Arduino UNO and table 2.2 shows the key parameters of the ATmega328 micro-controller, and figure 2.2 shows the pin diagram of the system.

2.4 MPU6050

The MPU-6050 is the worlds first, and has, only 6-axis MotionTracking device made for low power, low cost, and high performance need of gadgets like tablets and wearable sensors.

The MPU-6050 combines a 3-axis accelerometer and a 3-axis gyroscope on one silicon die and also contains an onboard Digital Motion Processor (DMP) for processing complex 6-axis MotionFusion algorithms. The device can access any other sensor

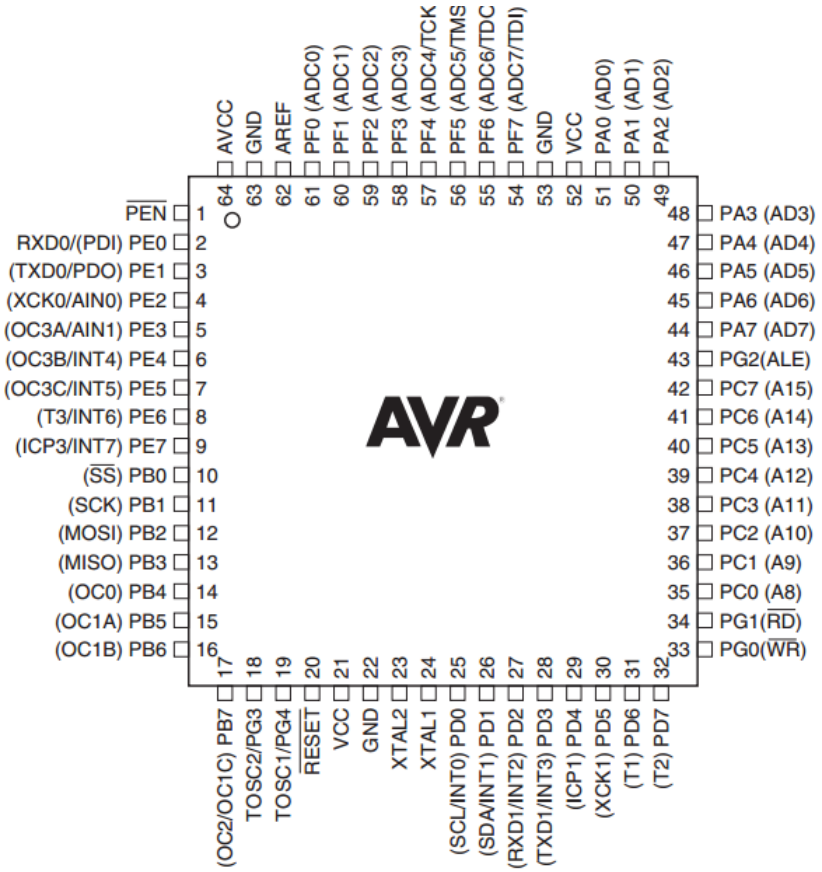

Figure 2.2.: PIN Layout of ATMega328

external to the device using the auxiliary IC bus interface, which helps reduce the load on the system processor. The MPU-6050 is packaged in 4x4x0.9 mm QFN packaging. Figure 2.3 shows the block diagram of the serial communication between a 3-axis compass, MPU6050, and an application processor. Figure 2.4 detail the connection between the Arduino UNO and the MPU6050 Inertial Measurement Unit.

The MPU-6050 has both fast and slow motions precision tracking. The device consists of a user-programmable gyro full-scale range of 250, 500, 1000, and 2000/sec (dps) and full-scale user-programmable accelerometer range of 2g, 4g, 8g, and 16g.

Table 2.2.: Key Parameters of ATMega328

Parameter	**Value**
Flash (Kbytes):	32 Kbytes
Pin Count:	32
Max. Operating Freq. (MHz):	20 MHz
CPU:	8-bit AVR
Max I/O Pins:	23
Ext Interrupts:	24
SPI:	2
TWI (I2C):	1
UART:	1
ADC channels:	8
ADC Resolution (bits):	10
ADC Speed (ksps):	15
Analog Comparators:	1

Figure 2.3.: MPU-6050 Serial Communication

2.5 ADLX345

ADXL345 is a digital accelerometer which is thin, light, and low power consumption device. The ADXL345 supports SPI (3 wire and 4 wire Communication) and I2C Serial Communication Interfaces. The acceleration data provided by the ADXL345 is 13-bit in resolution at $\pm 16g$ as two's compliment 16-bit data. The accelerometer is suitable for a mobile device as it requires very low power. The device can measure

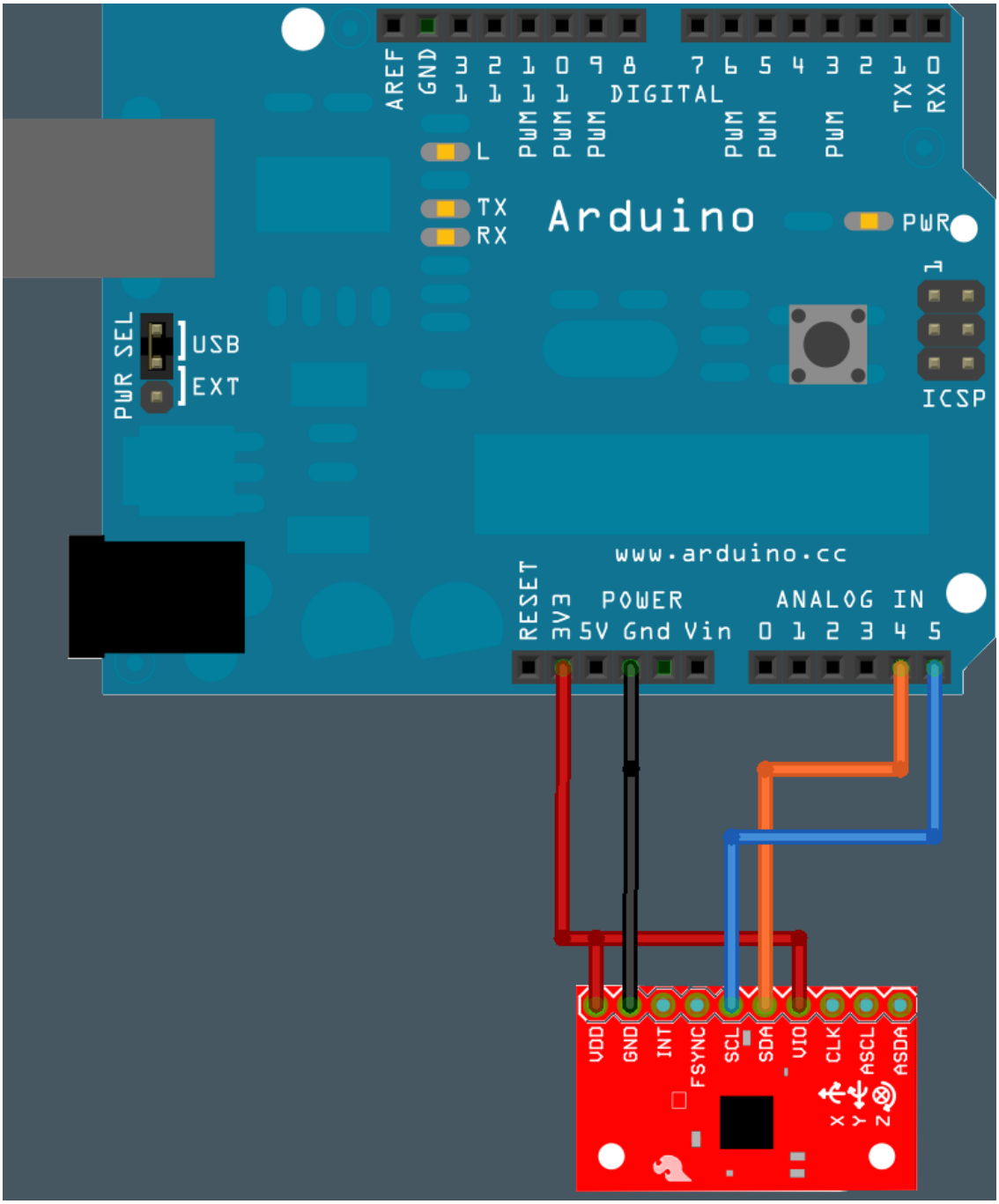

Figure 2.4.: MPU-6050 Arduino Connections

change in tilt with minimum changes as low as 1 degree, due to its high resolution of $4mg/LSB$ as well as the dynamic acceleration resulting from motion and shock.

The device provides Two-User programmable Interrupts which can be mapped to special functions like activity detection, free fall sensing, and tap sensing. The device provides a FIFO (First-in-First-out) buffer to reduce the host processor's work, and can be mapped to the interrupts. The Features of ADXL345 are:

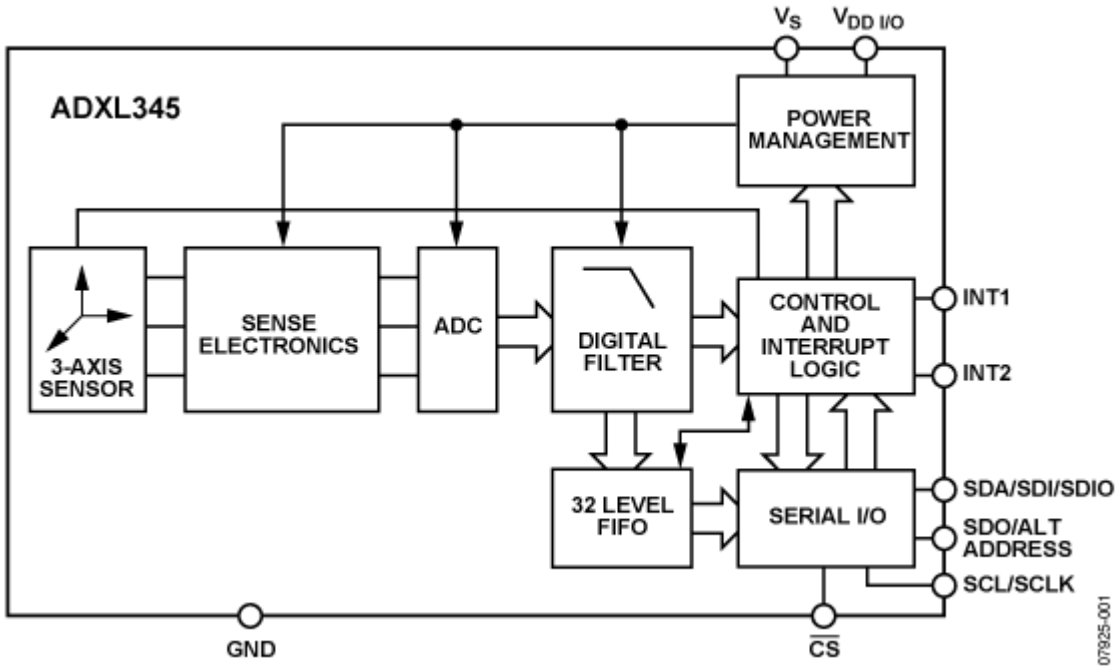

Figure 2.5.: Functional Block Diagram of ADXL345

- Ultra Low Power
- Power consumption scales automatically with bandwidth
- User-selectable resolution
- FIFO technology minimizes host processor load
- double tap detection
- Activity/inactivity monitoring

2.6 CC300 WiFi Module

CC3000 is a Wireless Network Processor designed by Texas Instruments which provides low power, low cost, and wireless access to the Internet. It is compatible with most low power MCU's and thus it became device of choice for embedded projects. The CC3000 has a complete TCP/IP stack, SPI communication interface running at

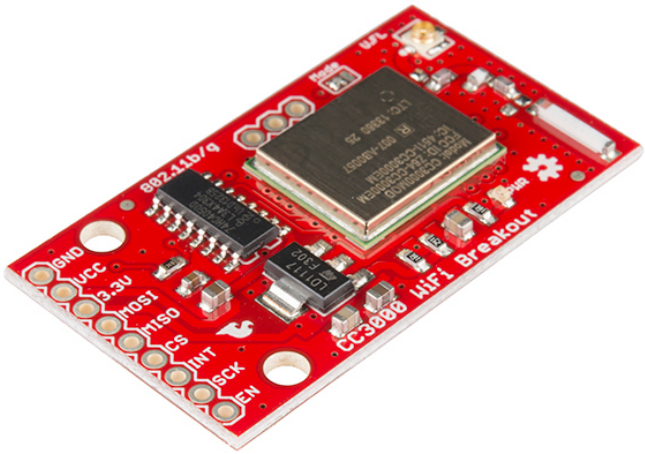

Figure 2.6.: CC3000 Breakout Board

$16MHz$ to communicate with the host processor, and compliant with IEEE 802.11 b/g, WEP, WPA/WPA2 security modes, and an on-board WiMAX Antenna. The CC3000 breakout board and the architecture of the API are shown in figures 2.6 and 2.7 respectively.

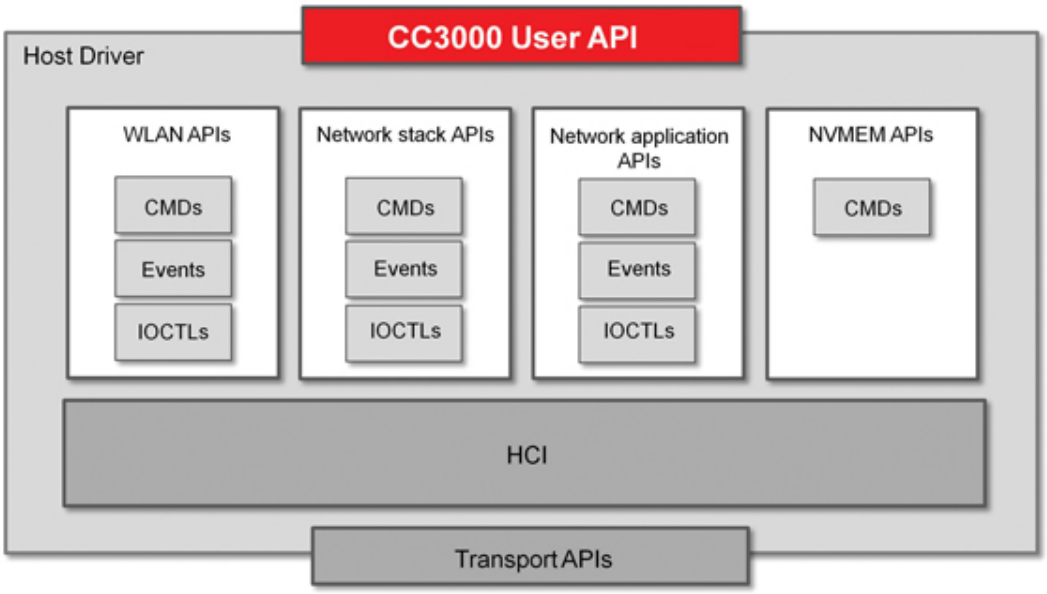

Figure 2.7.: CC3000 user API

3. SYSTEM DESIGN

3.1 Hardware Design

The Hardware model consists of an ATmega328P working at the center of the system. The MPU6050 and the ADXL345 communicate with the ATmega328P with I2C at the default speed of 100 kHz.

The accelerometer in MPU6050 is configured to work in the range of 8g and the gyroscope is configured to work in the range of 250 degrees/s. The ADXL345 is also configured to work in the range of 8g.

The IMU (Inertial Measurement Unit) and the accelerometer both have ADCs (Analog to Digital Converters) and the sampling rate for the ADC in ADXL345 is kept at 50 Hz as it results in lower power consumption and fulfills the requirements. The minimum sampling rate in MPU6050 is 1000 Hz for the accelerometer and gyroscope which is being used. Figure 3.1 gives the fall detection system design, and the schematic of the system is followed in figure 3.2.

3.2 Software Design

3.2.1 Initialization

The first step in the program loads the required libraries needed for the successful execution of the program. The libraries included Wire, I2Cdev, ADXL345, MPU6050, Kalman,SPI, SFE_CC3000, SFE_CC3000_Cliet, and math. The Wire library gives access to basic functions required for establishing communication using the I2C protocol. The I2Cdev library uses the functions in Wire library to build the read and write function for the I2C protocol. The ADXL345 library gives access to read and write functions which work on the registers of the ADXL345. The MPU6050

library provides these functions for the IMU.

The Kalman library provides the basic architecture required to utilize the Kalman filter within the system. The SPI library is used to build communication between the CC3000 WiFi Module and the ATmega328P using the SPI (Serial Peripheral Interface) protocol. The SFE_CC3000 and SFE_CC3000_Cient library provide the basic TCP/IP stack access required to construct a successful WiFi channel. The math library is required to do complex mathematical computations.

3.2.2 Setup

The second step is the setup, in which the Macros, global variables and the configuration is performed. The class objects of Kalman, MSP6050, ADXL345 and CC3000 are initialized. There are four functions declared, those are, Kalman, getG, getG1 and workdone. Kalman function performs the calculation of the angles from the accelerometer and gyroscope data. The MPU6050 is configured to a full scale range of 2g for the accelerometer and 250 degrees/s for the gyroscope, the in-built filters are configured to 260 Hz for accelerometer and 256 Hz for the gyroscope with a sampling rate of 8 kHz.

The getG and the getG1 functions take the raw data values and convert them to gs. The workdone function performs major part of the calculation. The input to the workdone function are the g values from the getG and getG1 functions. The algorithm is shown in the form of a flow chart in figure 3.3.

3.2.3 Calculations

The Loop is the main function which calls the other function and keeps running infinitely. The MPU6050 works in different configurations in the loop function as it is configured to a full scale range of 8g as per the requirements.

The sampling rate for the accelerometer and the gyroscope is 1 kHz. To perform this function the gyroscope configuration (GYRO_CONFIG) register and the

accelerometer configuration (ACCEL_CONFIG) register values have to be changed. The 4th and the 3rd bit of both GYRO_CONFIG and ACCEL_CONFIG i.e. FS_SEL [1:0] and AFS_SEL [1:0] are changed to decimal value 0 and 2 respectively. The same changes are made to the ADXL345 by using the setRange() function.

A delay is added in the next instruction for the accelerometers to stabilize as the range is being changed in every loop. After the accelerometers have stabilized the acceleration and gyroscope data is read from the device registers, these values are passed to the Kalman() function, which gives the angle of the body in two dimensions and the workdone function which calculates the features. If the thresholds are met, the data is passed to the postdate() function which posts the data on the server.

3.3 Design For Test and Data logging

The hardware design for the test and data logging involves an SD card shield from SparkFun electronics which is compatible with the Arduino UNO. The SD Card Shield uses SPI (Serial Peripheral Interface) to communicate with the Arduino UNO. The breakout board for the SD card is shown in figure 3.4. The accelerometer and gyroscope readings from the Arduino UNO are sent to the SD Card after computing the features and storing them onto the SD card for further analysis. The SD Shield requires the SPI and the SD libraries to send the data from the Arduino to the SD card as in figure 3.5. The microSD card implements a FAT16 or FAT32 file system which limits the file's size but may be enough for text data logging.

3.3.1 Datalogging Process

The SD card can log data at the sampling rate and save it for future analysis in different formats such as text, and .csv (Comma Separated Values) file format. The SD library provides many methods to work on the files in the SD card. The data from the accelerometer and the gyroscope is collected through the I2C interface and is in the raw format. In the raw form, the accelerometer, and gyroscope readings

give the values of acceleration and angular velocities from all three axes respectively. These readings are used to extract the features required for the algorithm and store them in the SD card. There is another variable "FALL", which stores the status of the fall occurrence. The "FALL" variable is ZERO if no fall is detected and ONE if fall is detected. The angle information for the two axes are also recorded along with the other features. Figure 3.6 and 3.7 detail the test system and the data logging algorithm respectively.

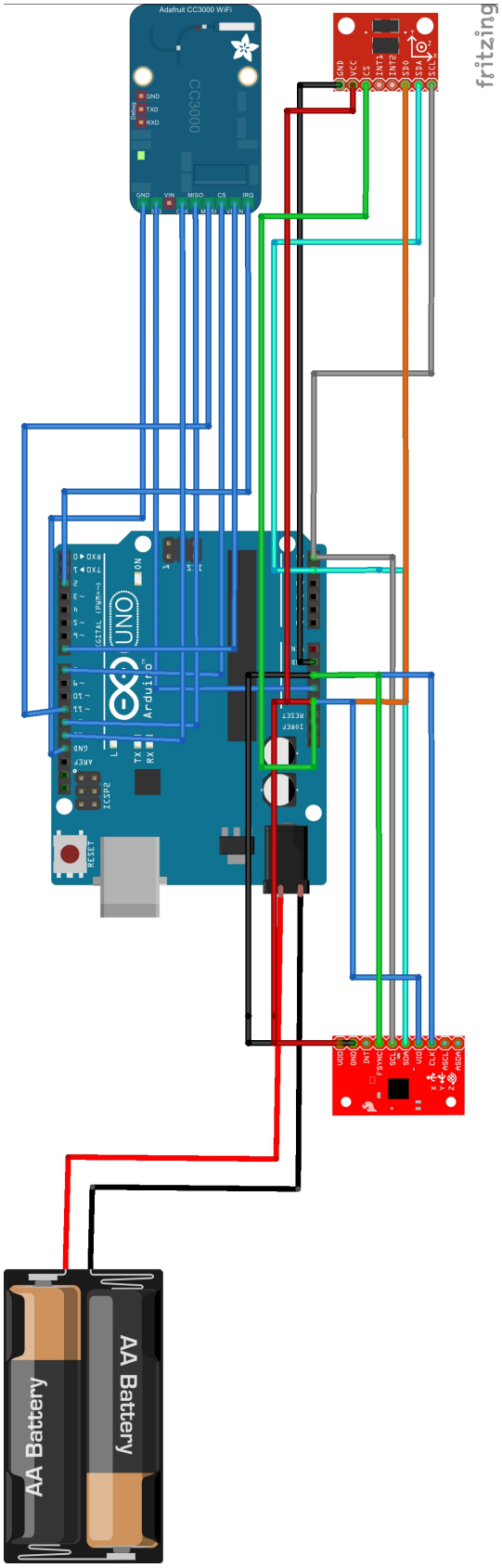

Figure 3.1.: Fall Detection System Design

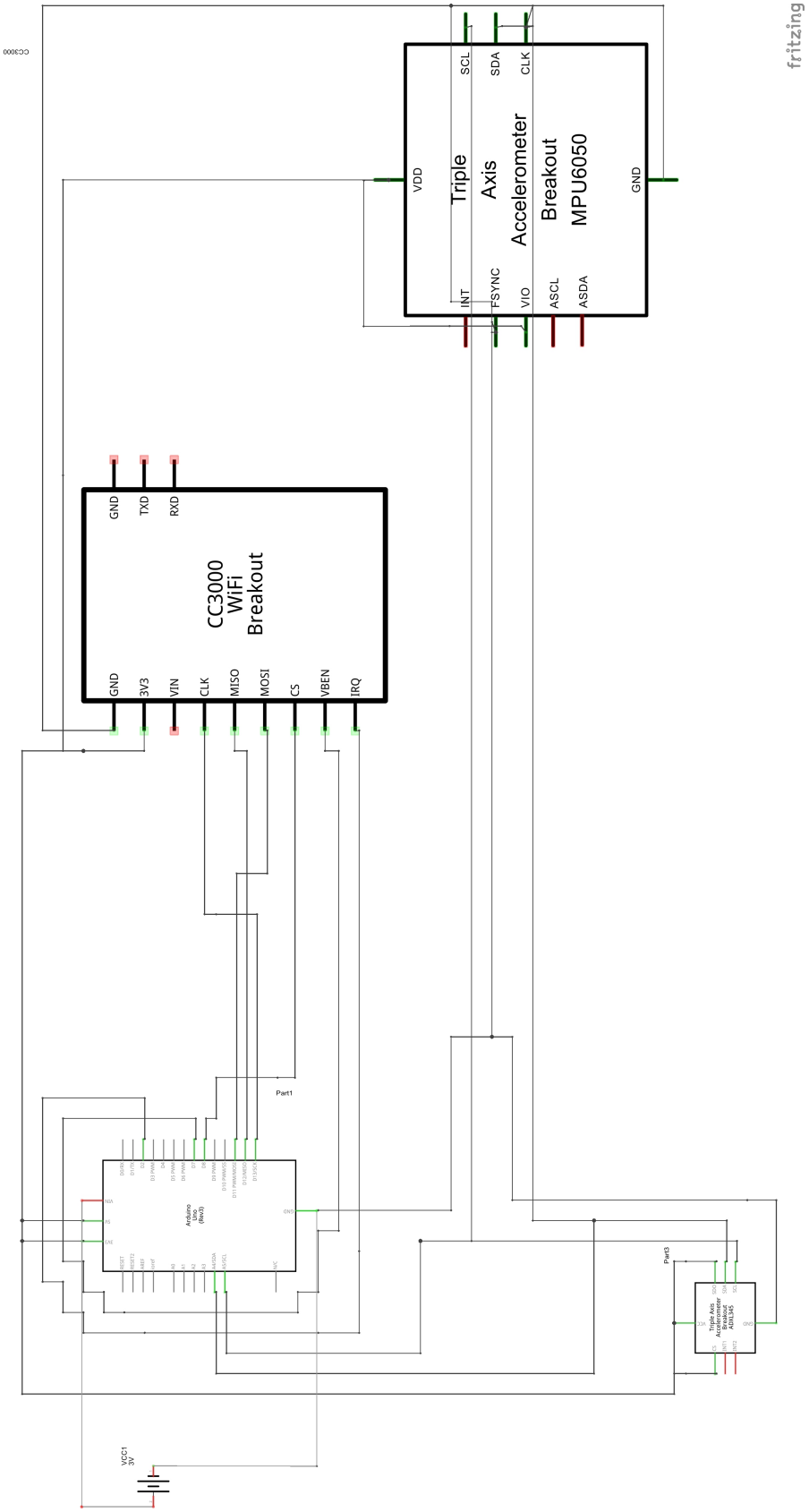

Figure 3.2.: Fall Detection System Schematic

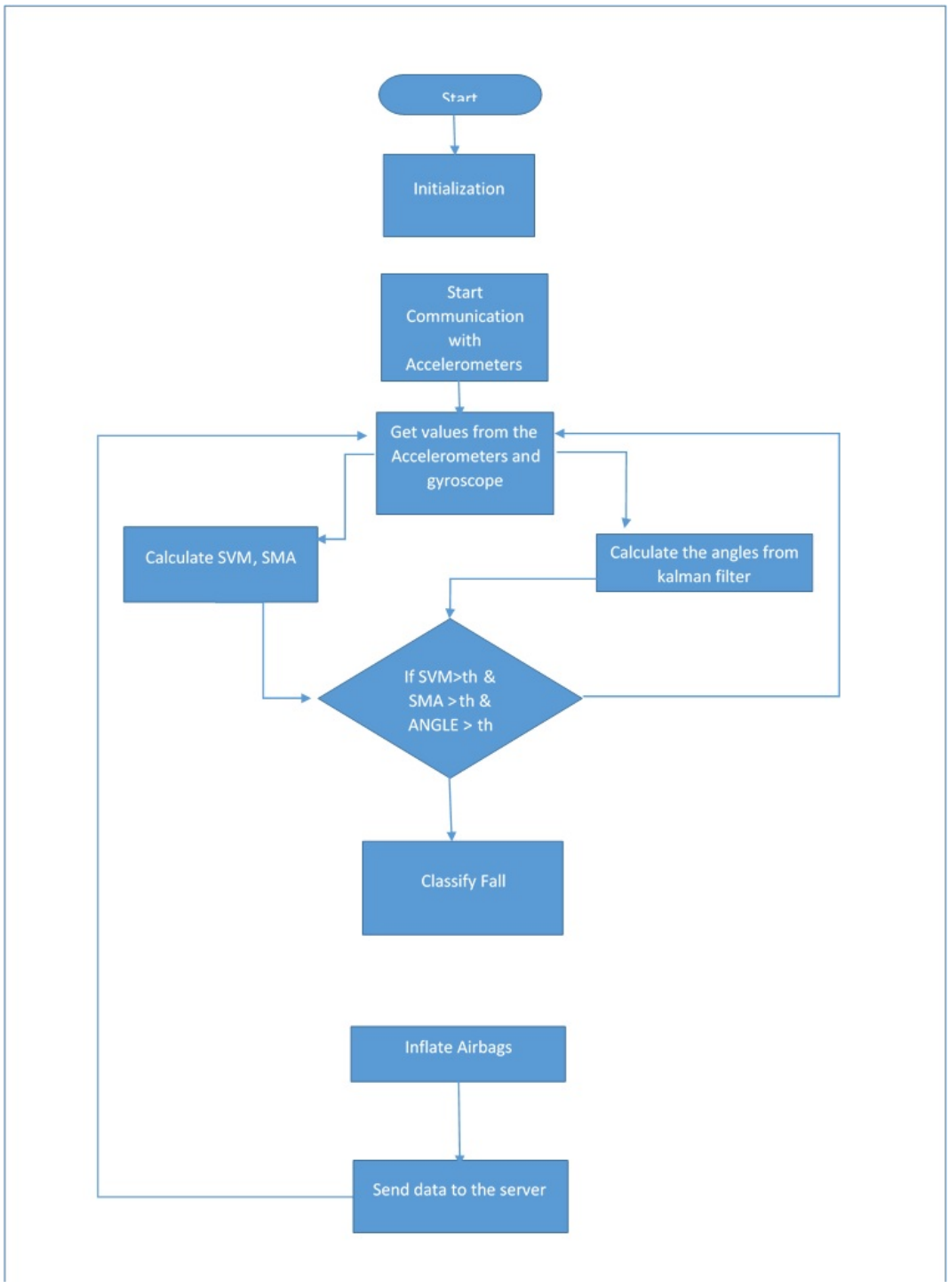

Figure 3.3.: Fall Detection Flow Chart

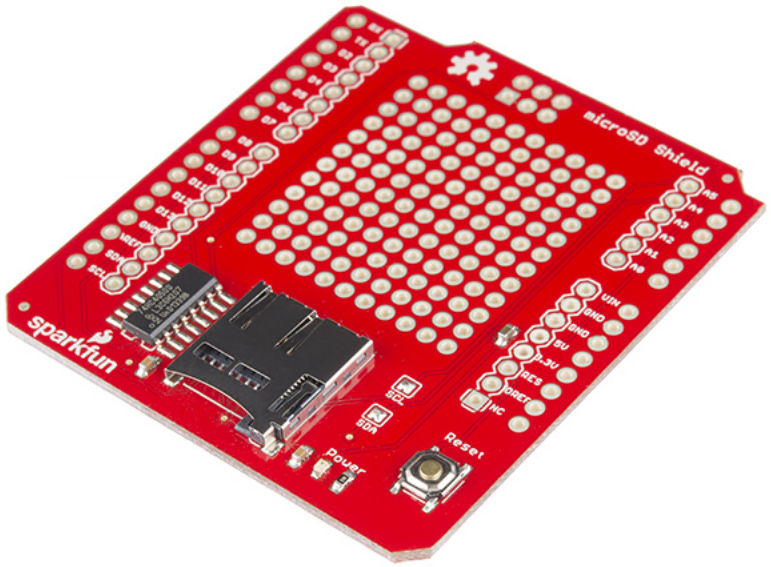

Figure 3.4.: SD Shield for Data logging

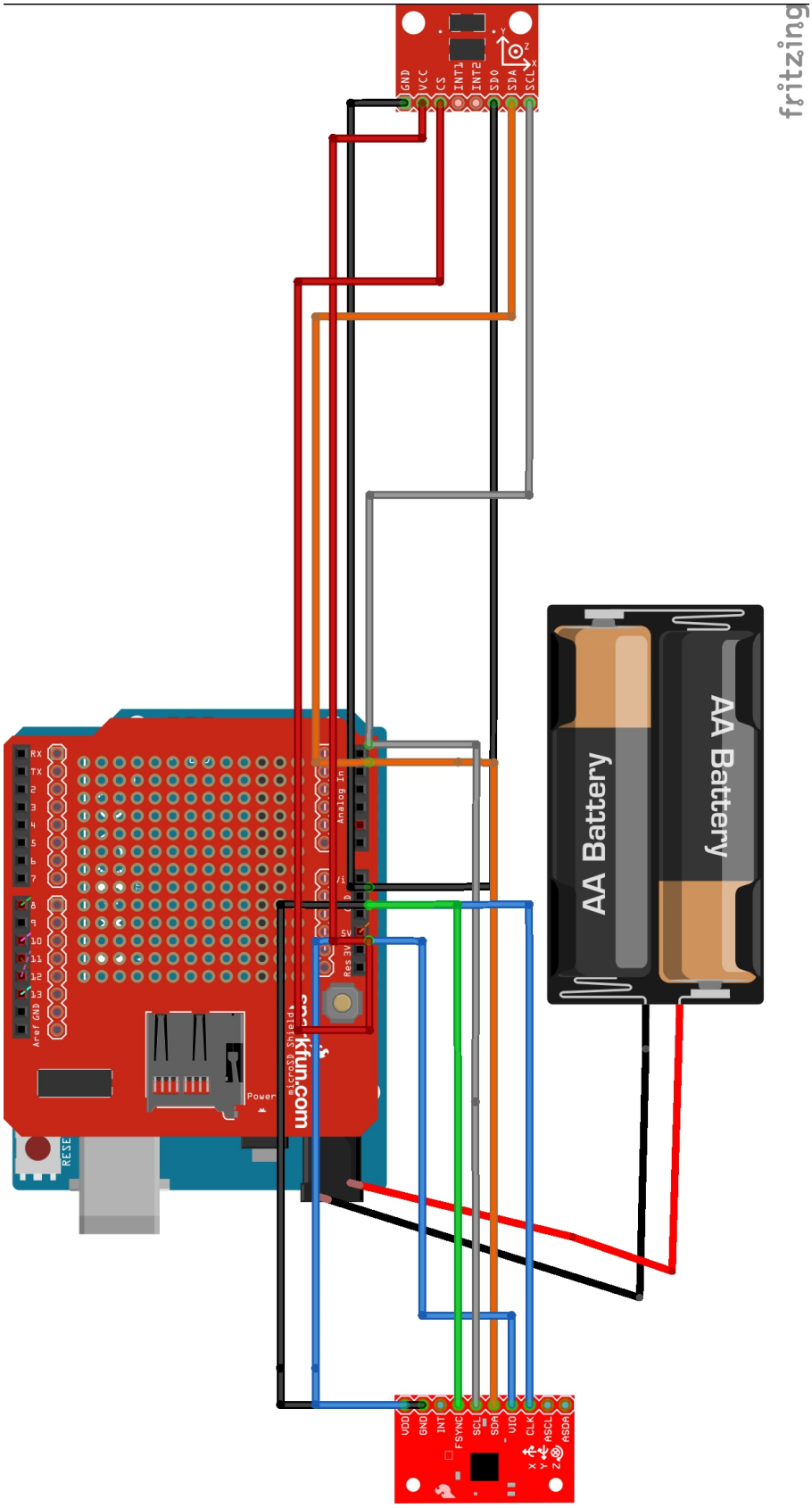

Figure 3.5.: System Design for Data logging and testing

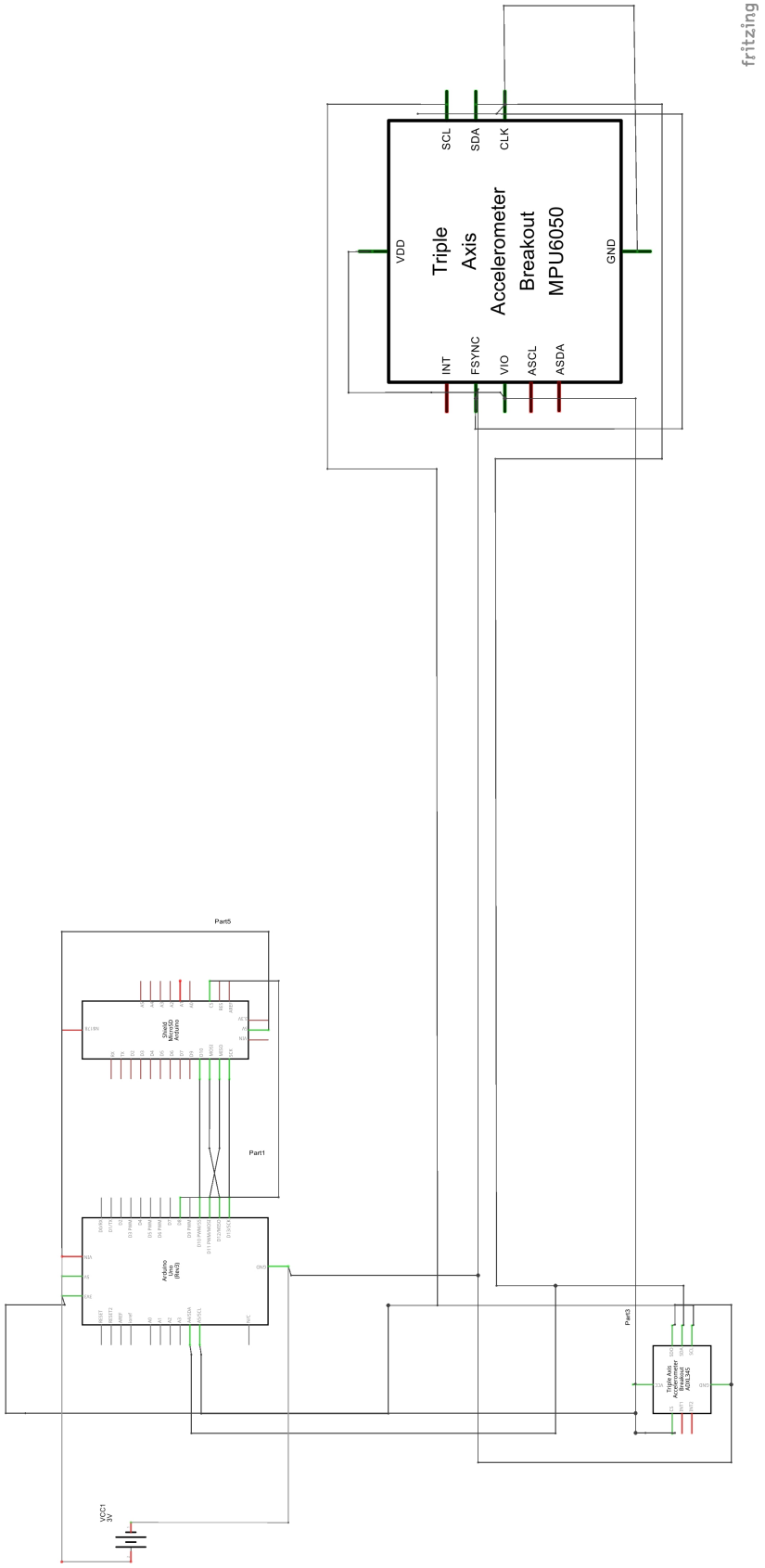

Figure 3.6.: Schematic of the Test System

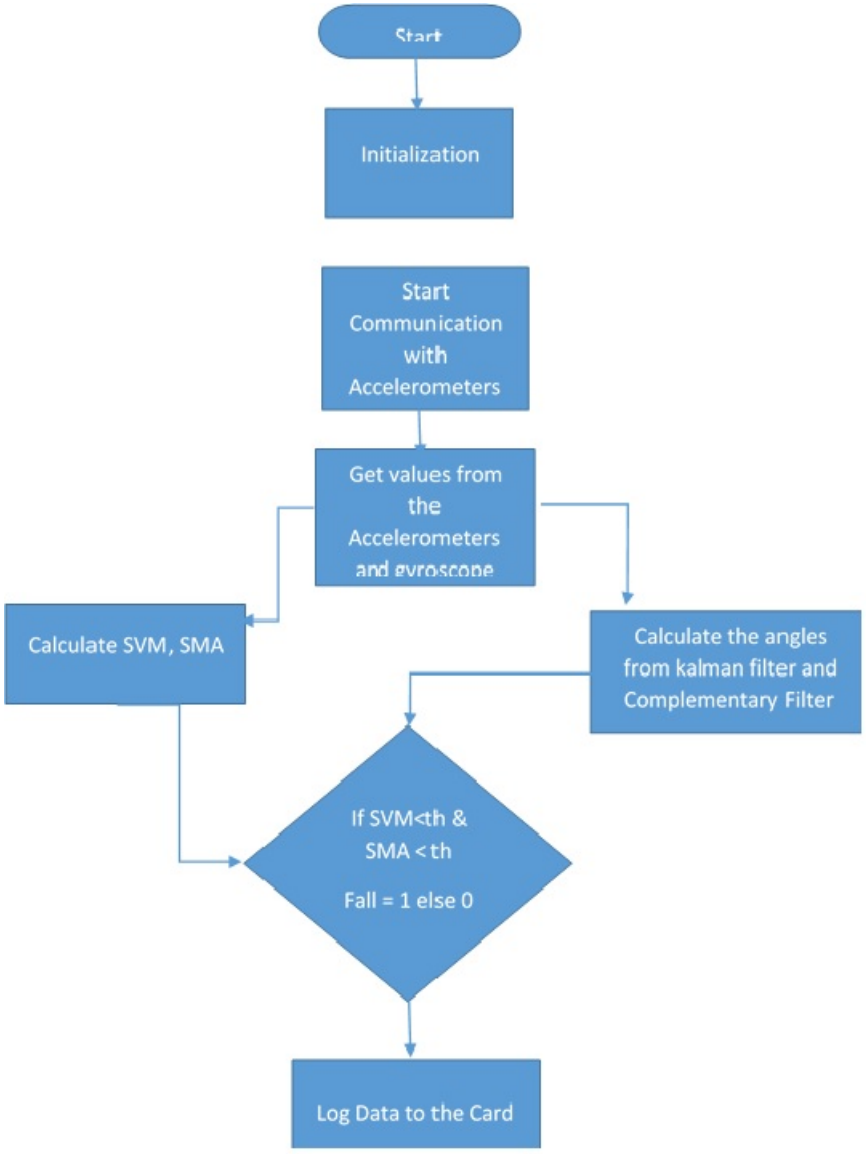

Figure 3.7.: Flow chart for testing and data logging system

4. COMPUTATIONAL MODEL

4.1 Feature Extraction

4.1.1 Acceleration

Acceleration is a vector quantity which holds the direction and magnitude. Accelerometer calculates the acceleration in three dimensions with the acceleration vector containing the direction and the magnitude as the length of the vector. The acceleration from the accelerometer is the addition of two components, one is the gravitational acceleration, and the second is the body acceleration. For further computational analysis these components have to be separated as BA (Body Acceleration) and GA (Gravity Acceleration) components according to [24]. Below are the equations used to calculate the angle between the acceleration vector and the three axes.

$$A_x = \arctan(\frac{x}{\sqrt{y^2 + z^2}}) \tag{4.1}$$

$$A_y = \arctan(\frac{y}{\sqrt{x^2 + z^2}}) \tag{4.2}$$

$$A_z = \arctan(\frac{z}{\sqrt{x^2 + y^2}}) \tag{4.3}$$

The angles are calculated based on the gravitational acceleration frame of reference as the accelerometer reading in the static state gives the magnitude of the gravitational vector. The accuracy of the angles in static or dynamic states will be uncertain as the acceleration vector will be influenced by the body acceleration.

4.1.2 Angular Velocity

The gyroscope provides the angular velocity (A_v) which is the derivative of the angular position (A_p).

$$Av = \frac{dAp}{dt} \tag{4.4}$$

The angular position can be obtained by taking an integration of the angular velocity over time.

$$Ap(t) = \int_0^t Av(t)dt \tag{4.5}$$

The Roll and Pitch can be computed by using the above formula, integrating over the period (T_s), and giving the angular movement in 2D. The computational formula for the Pitch (ϕ) and Roll (θ) measurement are below.

$$\theta = \sum_0^t Av_x(t)T_s \tag{4.6}$$

$$\phi = \sum_0^t Av_y(t)T_s \tag{4.7}$$

4.1.3 Signal Vector Magnitude (SVM)

The Signal Vector Magnitude (SVM) is the quantity of the acceleration vector. The fundamental assumption to check for fall is the change in the magnitude of acceleration as the human subject's body acceleration vector will be in the direction of the gravity vector, resulting in a rapid decrease in magnitude during the fall and before the impact with ground at which the quantity will peak. This feature is calculated according to Karatonis et al. [24]

$$SVM_i = \sqrt{x_i^2 + y_i^2 + z_i^2} \tag{4.8}$$

The SVM_i is the i^{th} sample calculated using the i^{th} samples of the acceleration values on the x, y and z-axis.

4.1.4 Signal Magnitude Area (SMA)

The Signal Magnitude Area (SMA) is another feature required to determine the fall. The quantity is the integration of the three varying components of the acceleration vector averaged over the sampling period summed together to clearly describe the static position and the dynamic position of the patient [24].

$$SMA = \frac{1}{t}(\int_0^t x(t)dt + \int_0^t y(t)dt + \int_0^t z(t)dt) \tag{4.9}$$

The x(t), y(t) and z(t) are the acceleration measurements in the three axes.

4.2 Complimentary Filter and Kalman Filter

The output from the accelerometer and the gyroscope have errors and give more accurate outputs in different scenarios. The data from these sensors have to be fused in order to get a proper angular position.

4.2.1 Inaccuracies in Accelerometers

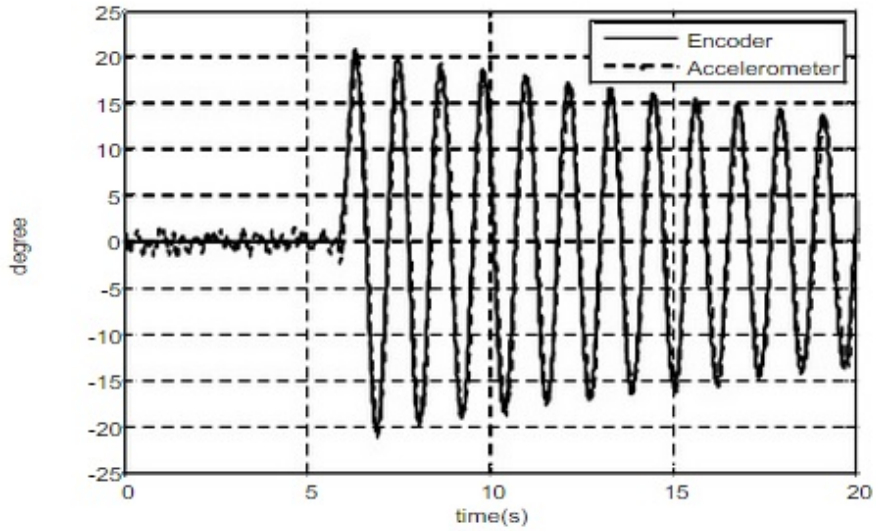

Figure 4.1.: Angle output without translational motion

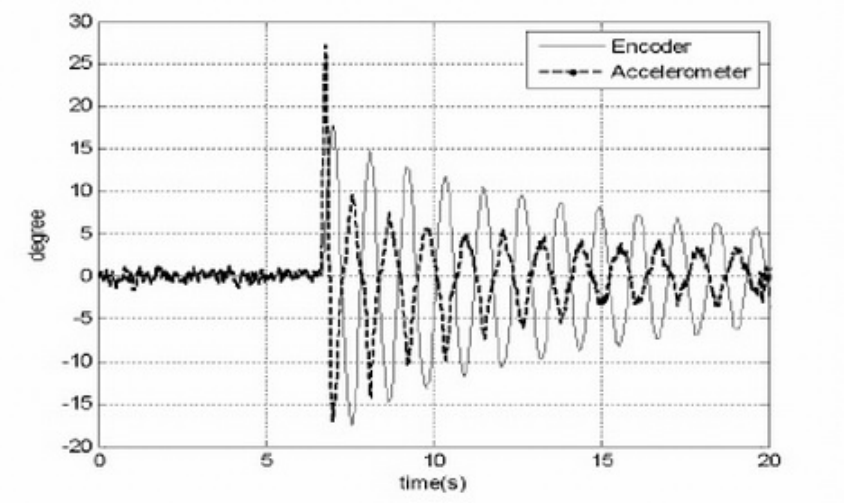

Figure 4.2.: Angle output with translational motion

The angle measured using an accelerometer is accurate until there are no other accelerations acting on the accelerometer other than the gravitational field. In the pre-Fall detection system, the acceleration due to the body is also accounted for by the accelerometer adding another part of translational motion to the total acceleration thus giving out inaccurate output [25]. According to [25], figure 4.2 and 4.3 show the angular measurements for two scenarios, one with no translational motion, and the other with translational motion. The result shows that the error is induced by the translational motion.

4.2.2 Inaccuracies in Gyroscopes

The gyroscopes raw data is the angular velocity which is converted to angular position by integrating the gyroscope output. There are many types of gyroscopes built using different technologies but the output has low or high amounts of bias error which also gets integrated when converting the angular velocity to angular position. This change in output gradually sums up, giving a drift in the angular as shown in figure 4.3. The [25] empirically showed this variation by comparing the encoder output with the integrated angular velocity from the gyroscope.

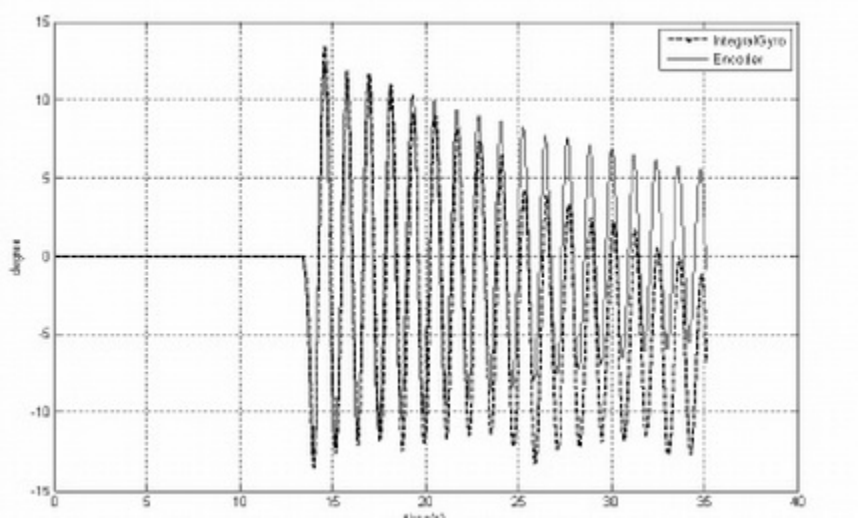

Figure 4.3.: Drift error due to Integration

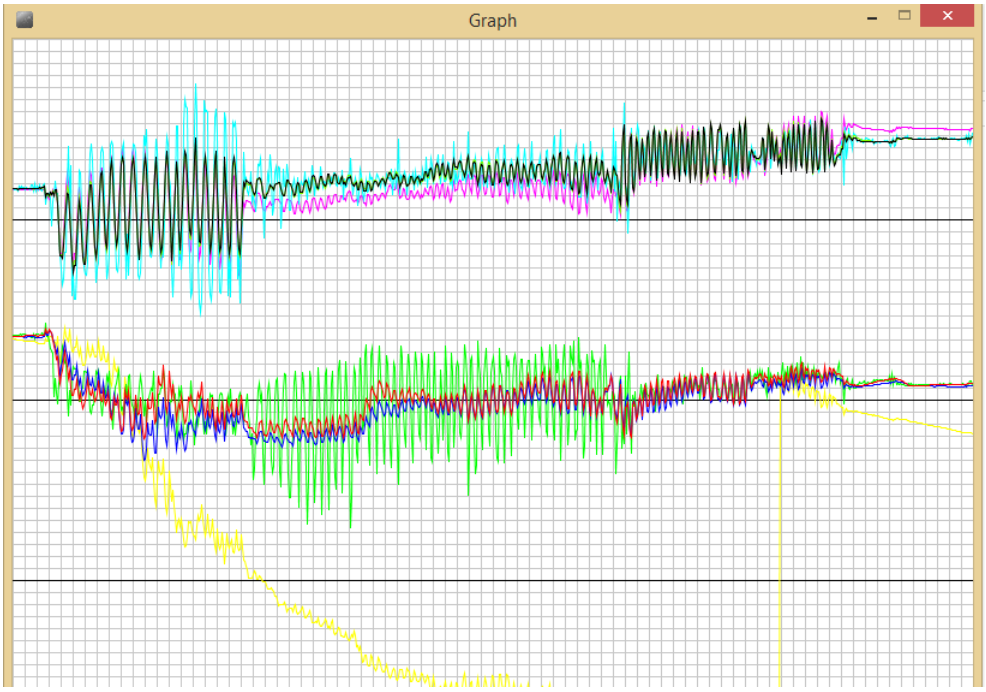

Figure 4.4.: Graph of the output from the Hardware

Figure 4.4 shows the output from the hardware. Which shows the angle calculated by using the different types of techniques discussed. The yellow line shows the drift of the output calculated using the gyroscope, and the lawn green line shows the

fluctuation of the output calculated by using the accelerometer. The red and the blue line in the graph are the outputs of the Kalman and the complimentary filter, which give a very similar output.

4.2.3 Complimentary Filter

The Complementary Filter combines the data from the accelerometer and the gyroscope and provides accurate results with minimum complexity and computational requirements as compared to Kalman filter. The ease of implementation on a small micro-controller with limited computational power makes it an ideal choice in the system.

The accelerometer gives more accurate when there are minimum number of forces other than the gravity acting on it. The performance of the accelerometer is best at the static or lower frequency. The gyroscope when in stationary position drifts due to the integration of errors and provide better results at higher frequencies thus performing complementary to the accelerometer. The complementary low pass filters, the accelerometer data, and high passes of the gyroscope data, have outputs multiplied with gains that are summing to unity.

$$G_g + G_a = 1 \tag{4.10}$$

where G_g is the gain for gyroscope and G_a is the gain for the accelerometer angle value. The gain for each quantity can be fine tuned depending on the application and the accuracy of the result. The complementary filter implementation is shown below with the gain values calculated by manual tuning.

$$compAX = 0.93 * (compAX + gXrate * dt) + 0.07 * roll \tag{4.11}$$

$$compAY = 0.93 * (compAY + gYrate * dt) + 0.07 * pitch \tag{4.12}$$

The angles from the gyroscope are calculated by integrating the output over time and the accelerometer angles, Roll and Pitch, are calculated by the formulas given in equations (4.11) and (4.12).

4.2.4 Kalman Filter

The Kalman filter is used in many high precision cases like aerospace, etc. The Kalman filter gives more accurate results than the complementary filter in most cases, but at the expense of complexity and computational inefficiency. The Kalamn filter in this research is used just for comparison and not in the prototype, as it is very difficult to implement in 8-bit micro-controller. Although the Kalman filter gives accurate results on the ATmega328P, the portability and the simplicity of the algorithm has to be maintained for general use.

5. THE PRACTICAL MODEL

5.1 Introduction

The system is designed in two configurations. One of the prototype consists of a Wifi breakout board from Sparkfun, which is connected to the serial peripheral interface bus, and provided a gateway to the Internet. The second prototype uses a SD card to store the data. Although both systems use the same sensors to gather inputs, the wearable fall detection system provide flexibility in terms of collecting data.

To reduce the hardware, data was collected by 5 wireless Z-star accelerometers, connected on the neck, stomach, back, knee, and calf. A software on the PC collected the data in an Excel sheet, to which the sensors sent the data. The sensors sent the data through Bluetooth which requires a lot of energy. As the sensors were operated on a battery, the sensor's range of communication reduces as the battery power is reduced. To make the process of collecting the data on the system itself, an SD card is installed. The system can be worn by a patient for a long duration and collect the data in real time. Figure 5.1 shows the format of data collected on the SD card.

The data from the Wifi can be sent to a server, local or remote. To show the functionality of the Wifi capability, the data is sent to an open-source remote server "Data.Sparkfun.com". The data can be sent to a computer with a server running on it. There are multiple open-source severs and databases available for building a practical application. The figure 5.2 shows an example of the data sent to the server.

	A	B	C	D	E	F	G	H	I
1	Time	SVM	SMA	SVM1	SMA1	Xangle	Yangle	Fall_Detected	
2	0	0.9	1.32	1.02	1.5	-11.65	21.44	0	
3	0.04	0.9	1.3	0.98	1.44	-11.77	21.64	0	
4	0.04	0.92	1.32	1	1.47	-11.56	21.7	0	
5	0.04	0.92	1.31	1.07	1.58	-11.12	21.7	0	
6	0.04	0.95	1.39	1.08	1.58	-10.46	21.85	0	
7	0.04	0.91	1.32	1.06	1.56	-9.69	21.86	0	
8	0.04	0.93	1.35	1.05	1.58	-8.82	22.28	0	
9	0.04	0.94	1.35	1.03	1.52	-7.99	22.63	0	
10	0.04	0.93	1.36	1.04	1.53	-7.25	22.86	0	
11	0.08	0.97	1.35	1.03	1.53	-6.65	23.59	0	
12	0.04	0.9	1.27	1.03	1.58	-6.62	23.81	0	
13	0.04	0.89	1.26	1.03	1.56	-6.67	23.93	0	
14	0.04	0.88	1.24	1.03	1.44	-6.65	24.09	0	
15	0.04	0.9	1.25	0.98	1.39	-6.77	24.43	0	
16	0.04	0.9	1.25	1.09	1.59	-7.32	25	0	
17	0.04	1.05	1.46	1.09	1.59	-8.03	25.95	0	
18	0.04	1	1.47	0.98	1.36	-8.56	26.42	0	
19	0.04	0.88	1.3	0.97	1.44	-8.89	26.16	0	
20	0.04	0.89	1.31	1.07	1.52	-8.86	25.92	0	
21	0.08	0.99	1.38	1.05	1.5	-8.49	26.82	0	

Figure 5.1.: Data in Excel sheet from the Hardware

5.2 The Portable System

The wearable fall detection system is a device worn by a patient, so it has to be designed to suit its purpose. To make the system comfortable, it should be in a form which makes it unobtrusive. After analyzing the data collect from 5 sensors at 5 different locations, 2 sensor location on the waist proved to be suitable for both, position and efficiency. This enabled the device to be designed in the form of a belt, making it comfortable and unnoticeable.

5.2.1 The Diagram

Figure 5.3 shows the design before the prototype was realized. The sensors are attached in the front and the back of the persons body. The sensor in the front provides the orientation information of the patient by using the data from the gyroscope and the accelerometer.

5.2.2 The Belt Device

Figure 5.4 shows the first prototype on the bread board, which has the Wifi module, and figure 5.5 shows the system produced in the form of a belt. The alternate design can be a System on Chip (SoC) which can be attached through clips on any belt or trouser. The wires in this systems case are not flexible but can be made adjustable for different sizes of patients. The system depends on the orientation of the body to make a decision, thus it is imperative to have the system placed properly on a patient. There is a high probability for the belt being oriented in a wrong way, due to the variation in size of the patient. The system solves this problem by calibrating itself to a zero position every time the system is worn.

5.3 Device Evaluation

5.3.1 Power Consumption

To improve the power consumption of the device, all design aspects are kept simple. The most simplistic approach to calculate the orientation is used, so as to implement it on a low power micro-controller. The sensors are configured to consume minimum power. The ADXl345 requires minimum power when it is configured at 50 hz, which is sufficient for the system to detect a fall.

5.3.2 High Speed

The complete duration of fall lies between 250 to 300 ms, and the system can detect a fall 160 to 200 ms before the impact. Considering the time required by the code to calculate the SVM, SMA, and the orientation, the system computes each value in 40 ms. This gives the system enough time to deploy any protection mechanism.

5.3.3 Complexity

The device consists of many components, each component should be efficient to make the system work properly. To fulfill the requirements the system should be simple and correct. To reduce the complexity of the system, many approaches have been analyzed and compared.

DATA.SPARKFUN.COM

Fall_Data Wifi demo for Fall Detection

Manage

JSON | CSV | MySQL | PostgreSQL | Atom

TAGS fall_detection

100% (50.00 of 50 MB) remaining.

sma	svm	timestamp	xangle	yangle
0	1	2015-04-10T20:57:24.856Z	-2	-61
0	1	2015-04-10T20:57:13.543Z	34	-16
0	1	2015-04-10T20:57:01.421Z	33	-19
0	1	2015-04-10T20:56:49.573Z	0	-27
0	1	2015-04-10T20:56:32.249Z	0	-24
0	0	2015-04-10T20:47:33.492Z	-2	-3
0	0	2015-04-10T20:47:18.077Z	-2	-3
0	0	2015-04-10T20:46:26.271Z	-2	-3
0	0	2015-04-10T20:46:06.876Z	-2	-3

Figure 5.2.: Wifi data on the server

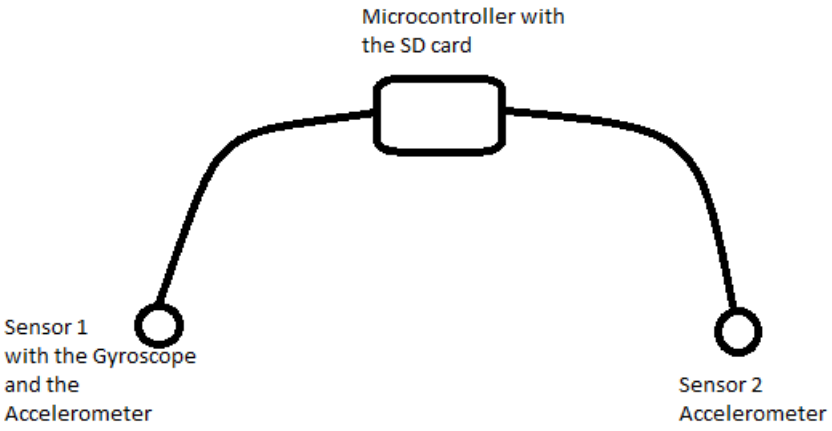

Figure 5.3.: Pre-realization design of the prototype

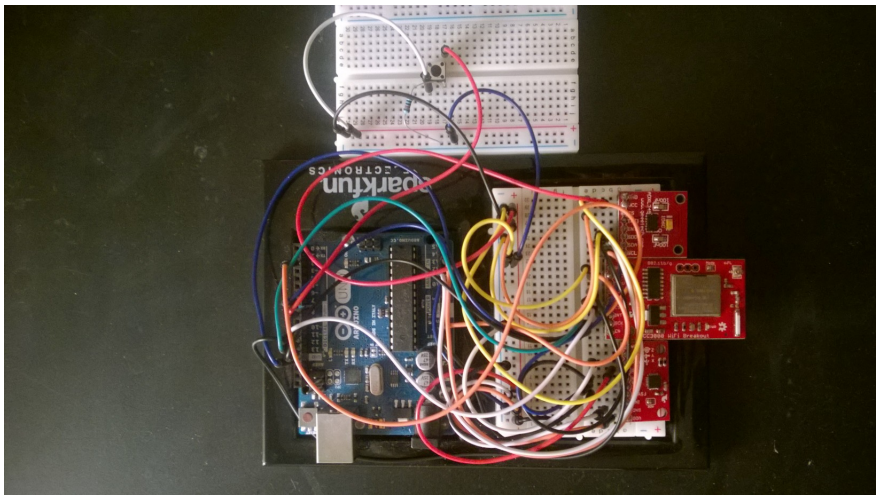

Figure 5.4.: First Prototype

Figure 5.5.: Final Prototype

6. DESIGN CONSIDERATIONS AND OPTIMIZATION

The Pre-Fall detection system is a wearable device meant to be worn through out the day by the patient supposedly in a monitored environment, thus the design of the wearable system should be comfortable, easy to wear and remove, robust, power efficient and accurate. The wearable system should be clutter free with least possible modules, providing enough computational power and adequate aspects of information required to give proper result.

6.1 Sensor Optimization

Data was collected using three sensors as shown in fig 6.2. The sensor placements was based on previous work using the thresholds calculated by the rattle software. Using the same threshold value the sensor combinations were used to find the optimum

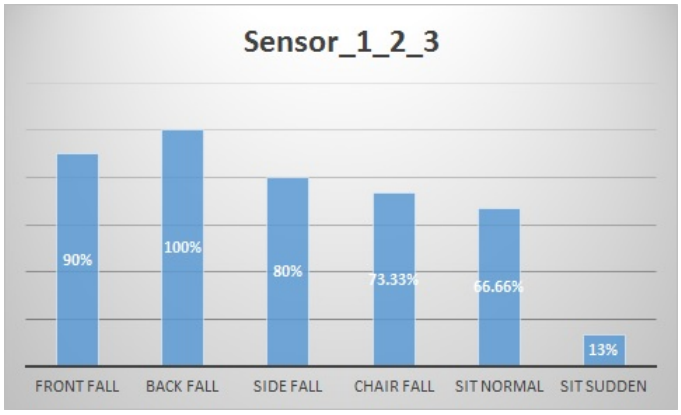

Figure 6.1.: Three Sensor Placement Accuracy Graph

pair of sensors and the best location for the placements of the sensors. The Sensors are numbered according to the fig 6.2. Sensors S4 and S5 are not taken into consideration due to the too much noise found in the output and the position on the body.

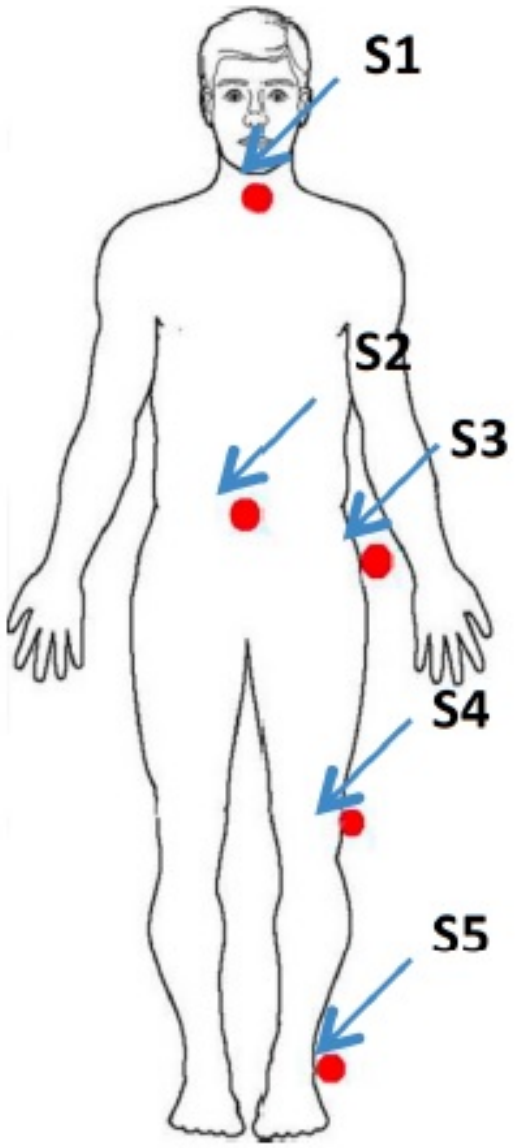

Figure 6.2.: Sensor Placement

The evaluation of the figures 6.3 and 6.4 provides very close results from the three sensors and the combination of the two sensors. Although sensors S1 and S3 provide the optimum accuracy for the placement, sensor S1 creates clutter and inconvenience and thus the Sensors S2 and S3 combination proves to be the best suitable placement acording to the design considerations.

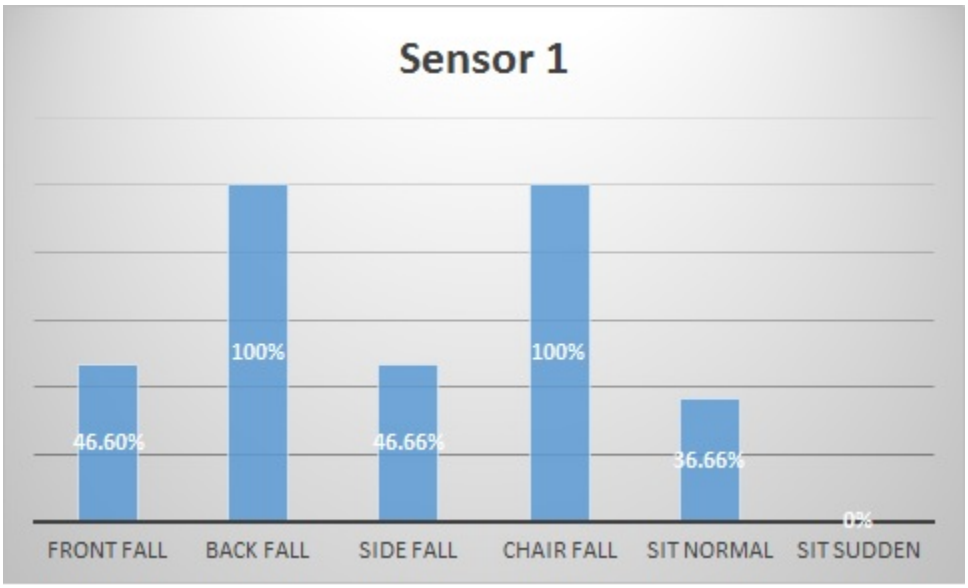

Figure 6.3.: Graph of accuracy of Sensor S1

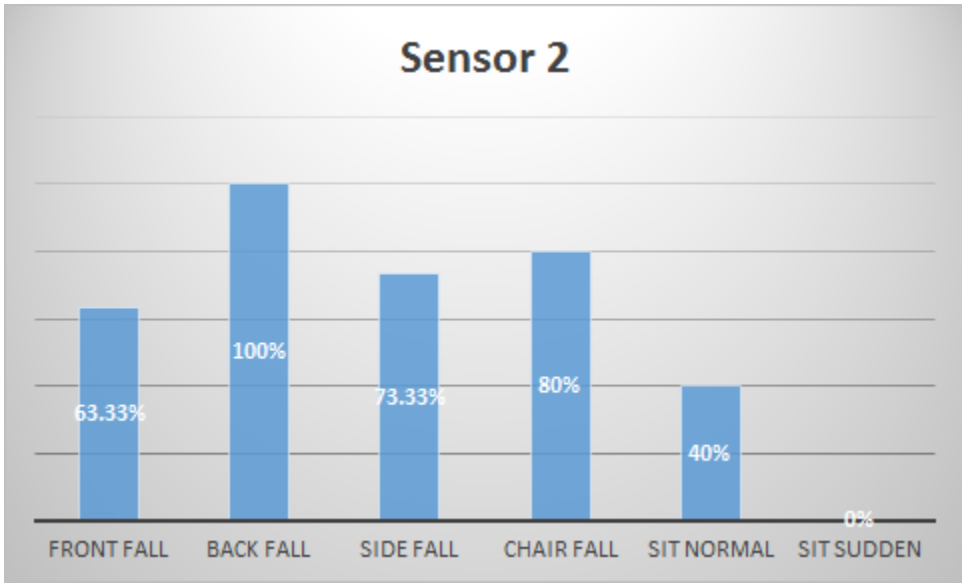

Figure 6.4.: Graph of Accuracy of Sensor S2

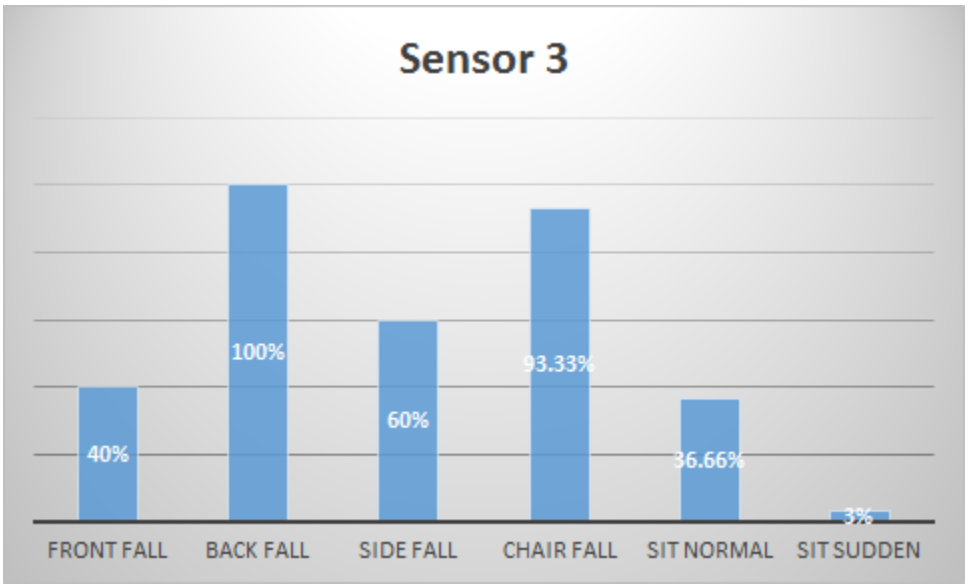

Figure 6.5.: Graph of Accuracy of Sensor S3

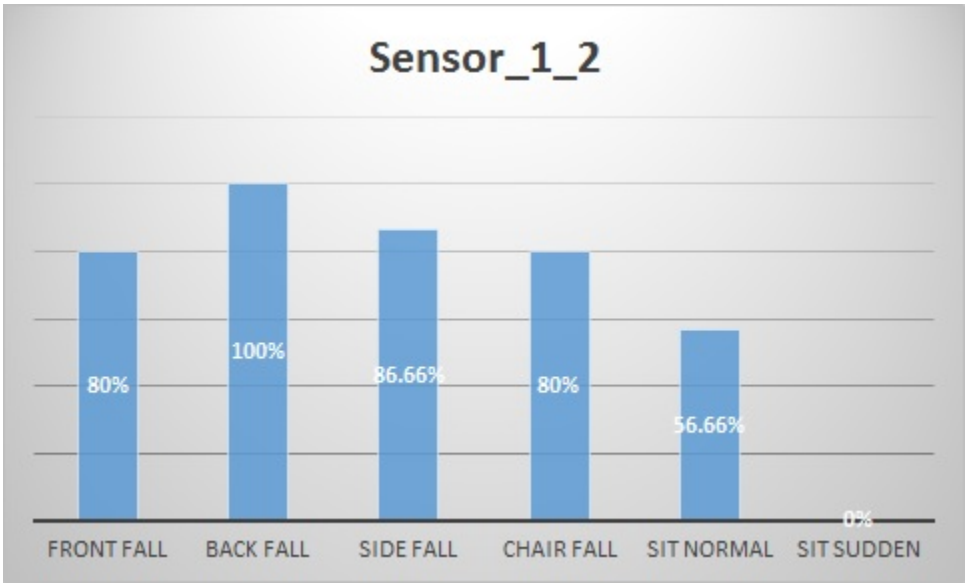

Figure 6.6.: Graph of Accuracy of Sensor Combination S1 and S2

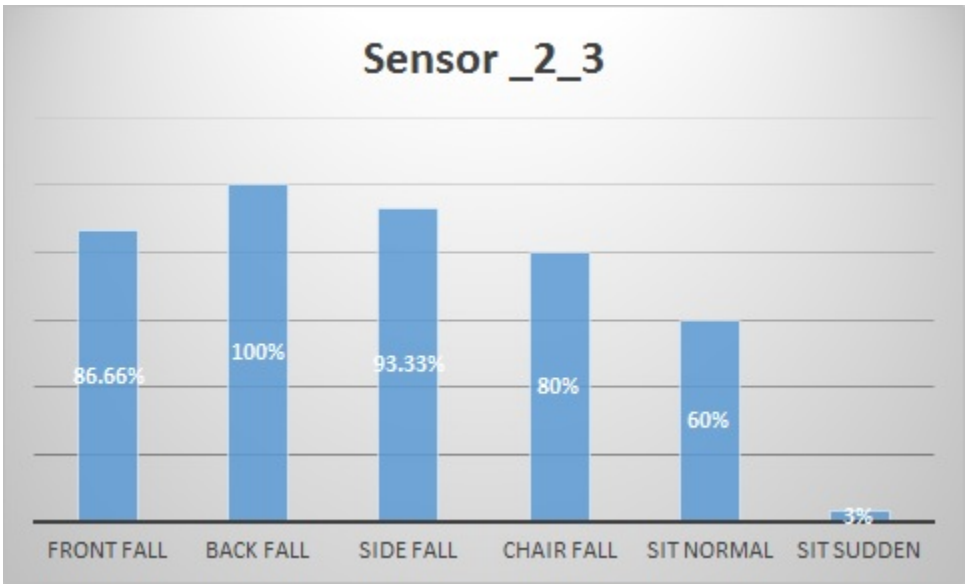

Figure 6.7.: Graph of Accuracy of Sensor Combination S2 and S3

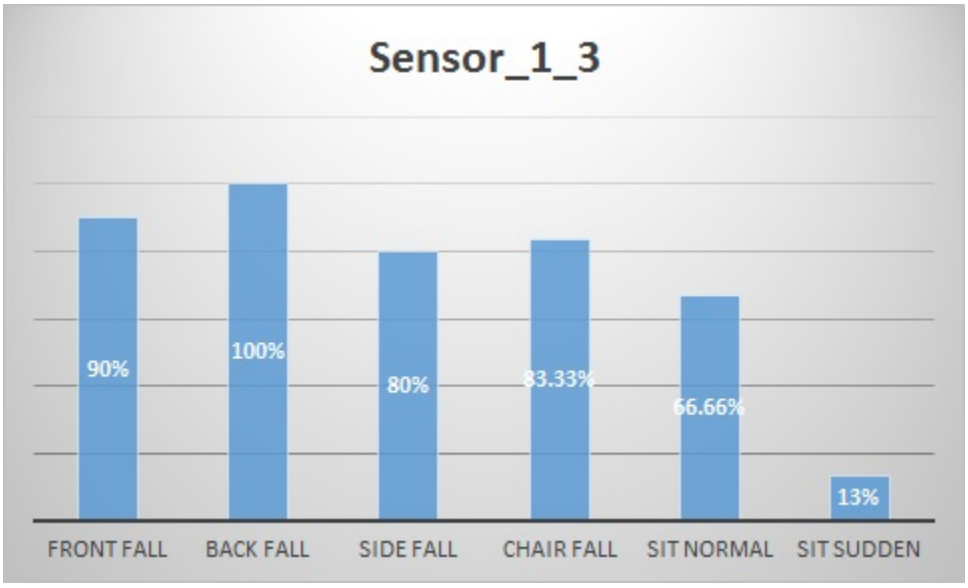

Figure 6.8.: Graph of Accuracy of Sensor Combination S1 and S3

7. RESULTS AND DISCUSSIONS

7.1 Test Results Without using Angle Thresholds

The Tests were performed on 4 male and 3 females. The protocols involved the four directions of fall i.e. Front, Back and sideways, the Activities of Daily living were chosen so as to cover the activities which come close to fall activities. The ADL(Activities of Daily Living) included are, siting down, standing up, walking and climbing up and down the stairs. Figures 7.1 and 7.2 give the sensitivity and specificity of the system.

7.2 Fall Pattern Detection

The System was able to detect the fall within 150 to 200ms before the impact occurs which gives enough time for the protection mechanism to react and provide protection. Figure 7.3 gives the pattern of the SMA and SVM values from the two accelerometers and shows the point where the fall is detected and the peak which is a point of impact of the subject with the ground. Each sample has a time difference of 40ms between each other.

7.3 Normal Activity Data results

Normal activities were performed which had the highest probability of false positives, like climbing down the stairs, climbing up the stairs, walking and sitting down. The complete set of activities were performed continually with the system worn. There were a very high number of false positives in the complete experiment mostly because of the very close threshold values to make sure there are no missed positives. The graph shows the points where the false positives were detected. Figures 7.4 and

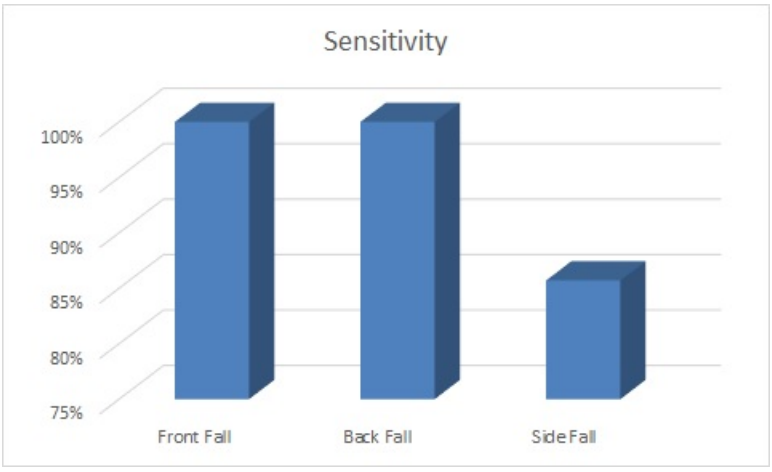

Figure 7.1.: Sensitivity of the System without Angle Threshold

Figure 7.2.: Specificity of the System without Angle Threshold

7.5 give the pattern with and without false positives.

Figure 7.5 shows the pattern with angle thresholds included in the algorithm. The output of the system is free from any false positives as the angle threshold provides a second check for the fall detection system, improving the overall performance of the system.

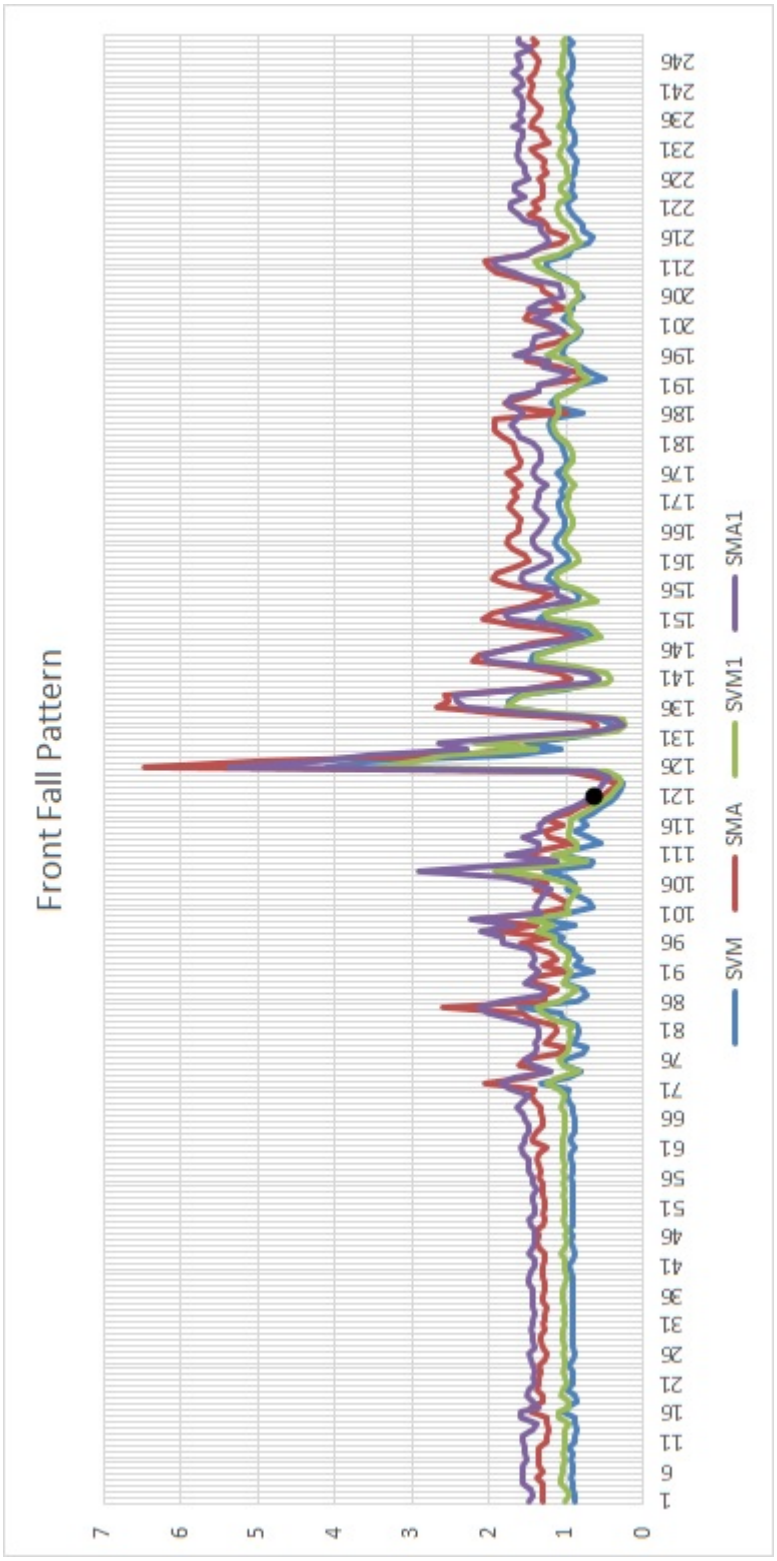

Figure 7.3.: Front fall Pattern Showing the detection Sample

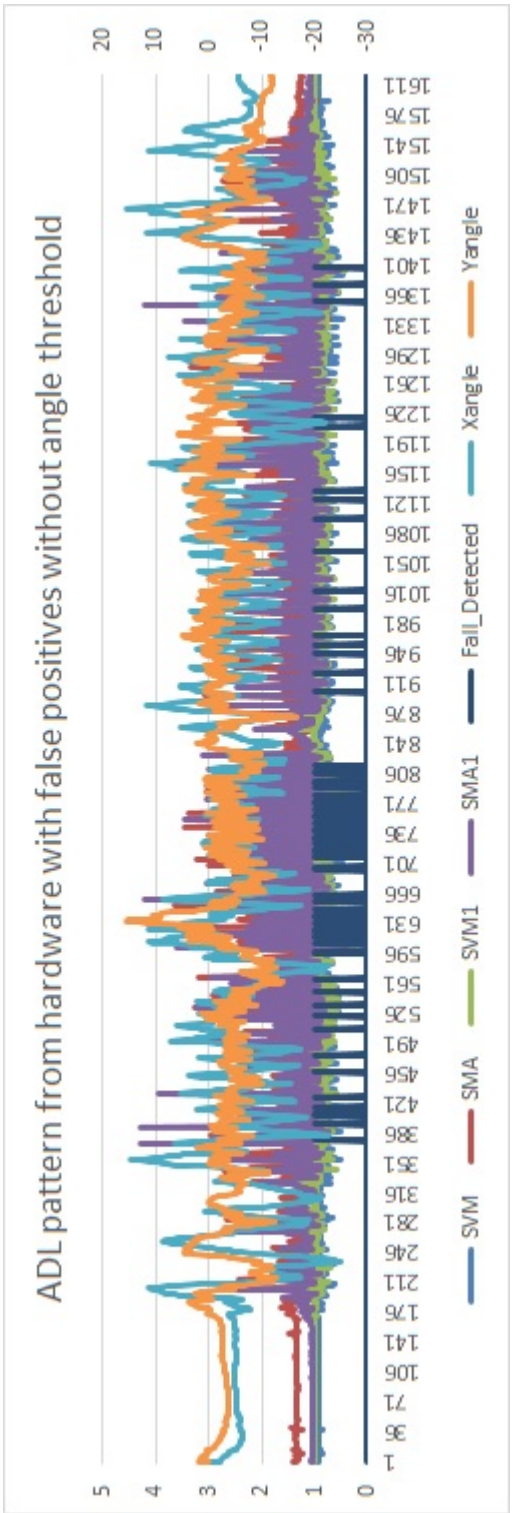

Figure 7.4.: ADL Pattern without Angle threshold with multiple False positives

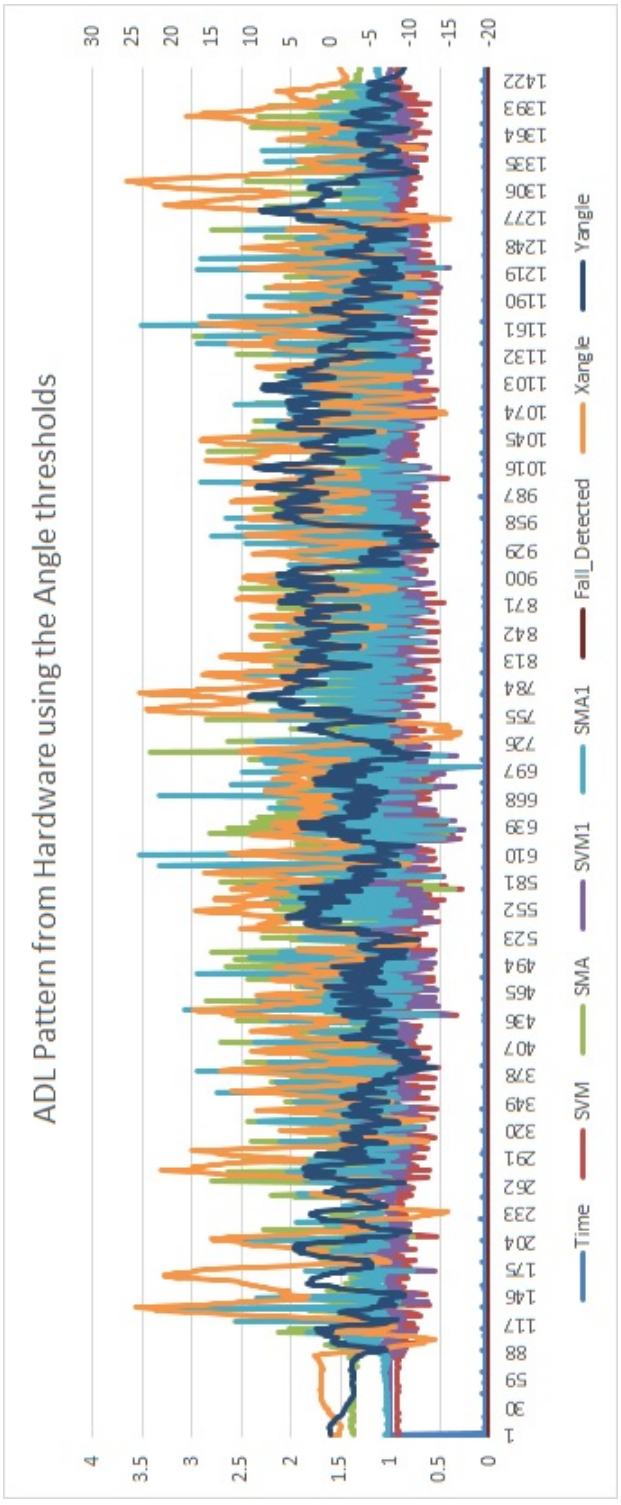

Figure 7.5.: ADL Pattern with Angle threshold without False positives

8. CONCLUSION

The designed system is minimalistic and practical and can be used commercially after further extensive testing. The system provides close to 100% specificity and sensitivity after the inclusion of the the second decision parameter which is the orientation of the body. The system also provides hardware and software framework for wireless connectivity which can be used in unlimited ways to develop applications.

As the work in an ongoing process, to further increase the efficiency of the system there is a need of a convenient way to gather data. The system provides the data logging capability with the SD card module attachment. The Software for the data logger provides data in the raw form and the calculated values which can be analyzed using any analysis tool as it gives the file in common CSV format.

9. FUTURE WORK

In order for the System to provides the hardware and software framework for web and mobile applications, an improved algorithm should consider more Physical Activity scenarios in order to increase the performance of the system. The addition of the WiFi module provides opportunity to connect and enhance the systems capability in unlimited ways.

The whole system can be built on a single SOC for improving the power efficiency, reducing size, and cost, and making the design more feasible to commercialization on a larger scale.

Further data collection and analysis can help reducing the number of sensors and help reduce the complexity and the cost of the system.

The data can be collected in a database which can then be used to provide information about the daily activity of the user, and can be displayed on a mobile application of the caretaker so as to be aware of the safety of the patient at all times.

REFERENCES

REFERENCES

[1] M. E. Tinetti and C. S. Williams, "Falls, injuries due to falls, and the risk of admission to a nursing home," *New England Journal of Medicine*, vol. 337, no. 18, pp. 1279–1284, 1997, pMID: 9345078.

[2] G. Wu and S. Xue, "Portable preimpact fall detector with inertial sensors," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 2, pp. 178–183, April 2008.

[3] S. Shan and T. Yuan, "A wearable pre-impact fall detector using feature selection and support vector machine," in *2010 IEEE 10th International Conference on Signal Processing (ICSP)*, October 2010, pp. 1686–1689.

[4] J. Liu and T. Lockhart, "Development and evaluation of a prior-to-impact fall event detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 7, pp. 2135–2140, July 2014.

[5] Y. He, Y. Li, and S.-D. Bao, "Fall detection by built-in tri-accelerometer of smartphone," in *2012 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, January 2012, pp. 184–187.

[6] S.-H. Fang, Y.-C. Liang, and K.-M. Chiu, "Developing a mobile phone-based fall detection system on android platform," in *Computing, Communications and Applications Conference (ComComAp), 2012*, January 2012, pp. 143–146.

[7] F. Sposaro and G. Tyson, "ifall: An android application for fall monitoring and response," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, September 2009, pp. 6119–6122.

[8] R. Tiwari, A. Singh, and S. Khan, "Using android platform to detect free fall," in *2013 International Conference on Information Systems and Computer Networks (ISCON)*, March 2013, pp. 161–163.

[9] J. Cheng, X. Chen, and M. Shen, "A framework for daily activity monitoring and fall detection based on surface electromyography and accelerometer signals," *IEEE Journal ofBiomedical and Health Informatics*, vol. 17, no. 1, pp. 38–45, January 2013.

[10] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, November 2008.

[11] S. H. Roy, M. Cheng, S.-S. Chang, J. Moore, G. De Luca, S. Nawab, and C. J. De Luca, "A combined semg and accelerometer system for monitoring functional activity in stroke," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 17, no. 6, pp. 585–594, 2009.

[12] C. Zhu and W. Sheng, "Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 3, pp. 569–573, 2011.

[13] T. Stiefmeier, G. Ogris, H. Junker, P. Lukowicz, and G. Troster, "Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario," in *2006 10th IEEE International Symposium on Wearable Computers*, October 2006, pp. 97–104.

[14] H. Ghasemzadeh, R. Jafari, and B. Prabhakaran, "A body sensor network with electromyogram and inertial sensors: Multimodal interpretation of muscular activities," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 2, pp. 198–206, March 2010.

[15] N. Vaswani, A. Roy-Chowdhury, and R. Chellappa, ""shape activity": a continuous-state hmm for moving/deforming shapes with application to abnormal activity detection," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1603–1616, October 2005.

[16] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[17] M. Popescu, Y. Li, M. Skubic, and M. Rantz, "An acoustic fall detector system that uses sound height information to reduce the false alarm rate," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, August 2008, pp. 4628–4631.

[18] J. Ward, P. Lukowicz, G. Troster, and T. Starner, "Activity recognition of assembly tasks using body-worn microphones and accelerometers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1553–1567, October 2006.

[19] R. Narasimhan, "Skin-contact sensor for automatic fall detection," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, August 2012, pp. 4038–4041.

[20] V. Bevilacqua, N. Nuzzolese, D. Barone, M. Pantaleo, M. Suma, D. D'Ambruoso, A. Volpe, C. Loconsole, and F. Stroppa, "Fall detection in indoor environment with kinect sensor," in *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, June 2014, pp. 319–324.

[21] M. Prado-Velasco, M. G. del Rio-Cidoncha, and R. Ortíz-Marín, "The inescapable smart impact detection system (isis): An ubiquitous and personalized fall detector based on a distributed divide and conquer strategy," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE, 2008, pp. 3332–3335.

[22] A. Bourke, P. W. J. Van de Ven, A. E. Chaya, G. M. OLaighin, and J. Nelson, "The design and development of a long-term fall detection system incorporated into a custom vest for the elderly." in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, August 2008, pp. 2836–2839.

[23] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomedical engineering online*, vol. 12, no. 1, p. 66, 2013.

[24] D. Karantonis, M. Narayanan, M. Mathie, N. Lovell, and B. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 156–167, January 2006.

[25] H. G. Min and E. T. Jeung, "Complementary filter design for angle estimation using mems accelerometer and gyroscope," 2006 ([Online]. Last Date Accessed: November 12, 2014), http://www.academia.edu/6261055/Complementary_Filter_Design_for_Angle_Estimation_using_MEMS_Accelerometer_and_Gyroscope.

APPENDIX

A. APPENDIX

A.1 Code for the Wifi Application

```
#include "Wire.h"
#include "I2Cdev.h"
#include "ADXL345.h"
#include "MPU6050.h"
#include "Kalman.h"
#include "math.h"
#include <SPI.h>
#include <SFE_CC3000.h>
#include <SFE_CC3000_Client.h>
#include <Progmem.h>

#define SMA_FORM(x, y, z)   (abs(x)   +  abs(y)  +  abs(z))
#define SVM_FORM(x, y, z)  (sqrt(    square(abs(x))    +
square(abs(y))  +  square(abs(z))    )  )
#define LED_PIN 13
#define CC3000_INT      2
#define CC3000_EN       7
#define CC3000_CS       10
#define IP_ADDR_LEN     4

ADXL345 accel;
MPU6050 accelgyro;
Kalman kalmanX;
```

```
Kalman kalmanY;

float divRange = 4096.0;
float divRange1 = 64.0;
float Gforce[3];
float Gforce1[3];
float SMA;
float SVM;
float SMA1;
float SVM1;
int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t axx, ayy, azz;
int valx, valy, valz;
int valx1, valy1, valz1;
double accX, accY, accZ;
double gyroX, gyroY, gyroZ;
double gyroXangle, gyroYangle;
double compAngleX, compAngleY;
double kalAngleX, kalAngleY;
uint32_t timer1;
uint8_t i2c_Data[14];
bool blinkState = false;
char ap_ssid[] = "NOKIA Lumia 920_1459";
char AP_password[] = "N4t89#55";
unsigned int AP_security = WLAN_SEC_WPA2;
unsigned int timeout = 30000;
char Server[] = "data.sparkfun.com";
```

```
SFE_CC3000 wifi =
 SFE_CC3000(CC3000_INT, CC3000_EN, CC3000_CS);
SFE_CC3000_Client client = SFE_CC3000_Client(wifi);


const String public_Key = "WGqyEmoZ75f6ynVAlxM1";
const String private_Key = "XRogBvDj47Foj8EXak9w";
const byte NUM_FIELDS = 4;
const String field_Names[NUM_FIELDS] =
{"sma", "svm", "xangle","yangle"};
String field_Data[NUM_FIELDS];
const int trigger_Pin = 8;
const int light_Pin = A0;
const int switch_Pin = 5;
String name = "Nikhil";
boolean newName = true;

void kalaman();
void getG();
void getG1();
void workdone();

void setup(){
    Wire.begin();
    Serial.begin(115200);
    setupWiFi();
  pinMode(triggerPin, INPUT_PULLUP);
  pinMode(switchPin, INPUT_PULLUP);
  pinMode(lightPin, INPUT_PULLUP);
```

```
    Serial.println("Initializing");
    accel.initialize();
    accelgyro.initialize();


    Serial.println("Testing");
    Serial.println(accel.testConnection()
    ? "ADXL345 successful" : " failed");
    Serial.println(accelgyro.testConnection()
     ? "MPU6050 success" : "MPU6050 fail");

    pinMode(LED_PIN, OUTPUT);

    delay(100);

  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  accX = ax;
  accY = ay;
  accZ = az;

   #ifdef RESTRICT_PITCH
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch =
   atan(-accX / sqrt(accY * accY + accZ * accZ))
  * RAD_TO_DEG;
#else // Eq. 28 and 29
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ))
```

```
    * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  kalmanX.setAngle(roll);
  kalmanY.setAngle(pitch);
  gyroXangle = roll;
  gyroYangle = pitch;
  compAngleX = roll;
  compAngleY = pitch;

   timer1 = micros();
 }

  void loop(){
  double elapse = 0;

  kalaman();
  accelgyro.setFullScaleAccelRange(MPU6050_ACCEL_FS_8);
  accel.setRange( ADXL345_RANGE_8G );
  delay(10);
  accel.getAcceleration(&axx, &ayy, &azz);
  valx1 = axx;
  valy1 = ayy;
  valz1 = azz;
  accelgyro.getAcceleration(&ax, &ay, &az);
  valx = ax;
  valy = ay;
  valz = az;
```

```
  workdone();

  if (!digitalRead(triggerPin))
  {
    // Gather data:
    fieldData[0] = String(int(SVM));
    fieldData[1] = String(int(SMA));
    fieldData[2] = String(int(kalAngleX));
    fieldData[3] = String(int(kalAngleY));
    // Post data:
    Serial.println("Posting!");
    postData();
    delay(10);
  }

  //elapse = micros()-timer1;
  //Serial.println(elapse);
  }



void kalaman ()
{
  // read raw accel measurements from device
  accelgyro.setFullScaleAccelRange(0);
  delay(10);
    accel.getAcceleration(&axx, &ayy, &azz);
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    accX = ax;
```

```
    accY = ay;
    accZ = az;
    gyroX = gx;
    gyroY = gy;
    gyroZ = gz;

  double dt = (double)(micros() - timer1) / 1000000;
  timer1 = micros();



#ifdef RESTRICT_PITCH
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch =
  atan(-accX / sqrt(accY * accY + accZ * accZ))
   * RAD_TO_DEG;
#else
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ))
   * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  double gyroXrate = gyroX / 131.0;
  double gyroYrate = gyroY / 131.0;

#ifdef RESTRICT_PITCH
  if ((roll < -90 && kalAngleX > 90) ||
  (roll > 90 && kalAngleX < -90)) {
    kalmanX.setAngle(roll);
    compAngleX = roll;
```

```
    kalAngleX = roll;
    gyroXangle = roll;
  } else
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);

  if (abs(kalAngleX) > 90)
    gyroYrate = -gyroYrate;
  kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
#else
  if ((pitch < -90 && kalAngleY > 90)
  || (pitch > 90 && kalAngleY < -90)) {
    kalmanY.setAngle(pitch);
    compAngleY = pitch;
    kalAngleY = pitch;
    gyroYangle = pitch;
  } else
    kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);

  if (abs(kalAngleY) > 90)
    gyroXrate = -gyroXrate;
  kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);
#endif

  gyroXangle += gyroXrate * dt;
  gyroYangle += gyroYrate * dt;
  //gyroXangle += kalmanX.getRate() * dt;
  //gyroYangle += kalmanY.getRate() * dt;

  compAngleX = 0.93 * (compAngleX + gyroXrate * dt)
```

```
 + 0.07 * roll;
 compAngleY = 0.93 * (compAngleY + gyroYrate * dt)
 + 0.07 * pitch;

 // Reset the gyro angle when it has drifted too much
 if (gyroXangle < -180 || gyroXangle > 180)
   gyroXangle = kalAngleX;
 if (gyroYangle < -180 || gyroYangle > 180)
   gyroYangle = kalAngleY;



   blinkState = !blinkState;
   digitalWrite(LED_PIN, blinkState);
    }



void workdone(){
 uint8_t rate;
rate = accelgyro.getFullScaleAccelRange();
//Serial.println(rate);
getG();
getG1();


      SMA1 = SMA_FORM(Gforce1[0], Gforce1[1], Gforce1[2]);
      SVM1 = SVM_FORM(Gforce1[0], Gforce1[1], Gforce1[2]);
      SMA = SMA_FORM(Gforce[0], Gforce[1], Gforce[2]);
      SVM = SVM_FORM(Gforce[0], Gforce[1], Gforce[2]);
```

```
    if (SVM1 <= 0.744 && SMA1 <= 0.9197
     && SVM <= 0.8182 && SMA <= 0.9297 ){
       Serial.println("fall detected");
       };
}

void getG(){

  Gforce[0] = valx/divRange;
  Gforce[1] = valy/divRange;
  Gforce[2] = valz/divRange;

}

 void getG1(){

  Gforce1[0] = valx1/divRange1;
  Gforce1[1] = valy1/divRange1;
  Gforce1[2] = valz1/divRange1;

}

void postData()
{
  if ( !client.connect(server, 80) )
  {
    // Error: 4 - Could not make a TCP connection
```

```
    Serial.println(F("Error: 4"));
  }


  client.print("GET /input/");
  client.print(publicKey);
  client.print("?private_key=");
  client.print(privateKey);
  for (int i=0; i<NUM_FIELDS; i++)
  {
    client.print("&");
    client.print(fieldNames[i]);
    client.print("=");
    client.print(fieldData[i]);
  }
  client.println(" HTTP/1.1");
  client.print("Host: ");
  client.println(server);
  client.println("Connection: close");
  client.println();

  while (client.connected())
  {
    if ( client.available() )
    {
      char c = client.read();
      Serial.print(c);
    }
  }
```

```
  Serial.println();
}


void setupWiFi()
{
  ConnectionInfo connection_info;
  int i;

  // Initialize CC3000 (configure SPI communications)
  if ( wifi.init() )
  {
    Serial.println(F("CC3000 Ready!"));
  }
  else
  {
    // Error: 0 - Something went wrong during CC3000 init!
    Serial.println(F("Error: 0"));
  }

  // Connect using DHCP
  Serial.print(F("Connecting to: "));
  Serial.println(ap_ssid);

  if(!wifi.connect
  (ap_ssid, ap_security, ap_password, timeout))
  {
    // Error: 1 - Could not connect to AP
    Serial.println("Error: 1");
```

```
  }
  Serial.println("Connected!!");
  // Gather connection details and print IP address
  if ( !wifi.getConnectionInfo(connection_info) )
  {
    // Error: 2 - Could not obtain connection details
    Serial.println(F("Error: 2"));
  }
  else
  {
    Serial.print(F("My IP: "));
    for (i = 0; i < IP_ADDR_LEN; i++)
    {
      Serial.print(connection_info.ip_address[i]);
      if ( i < IP_ADDR_LEN - 1 )
      {
        Serial.print(".");
      }
    }
    Serial.println();
  }
}
```

A.2 Code for the Data logger

```
#include "Wire.h"
#include "I2Cdev.h"
#include "ADXL345.h"
#include "MPU6050.h"
```

```
#include "Kalman.h"
#include "math.h"
#include <SD.h>

#define SMA_FORM(x, y, z)   (abs(x)    +  abs(y)  +  abs(z))
#define SVM_FORM(x, y, z)  (sqrt(    square(abs(x))
   +  square(abs(y))  +  square(abs(z))    )  )
#define OUTPUT_READABLE_ACCELGYRO
#define LED_PIN 13 // (Arduino is 13, Teensy is 6)

ADXL345 accel;
MPU6050 accelgyro;
Kalman kalmanX;
Kalman kalmanY;

int calsel =1;
int fall = 0;
int id;
int CS_pin = 8;
float divRange = 4096.0;
float divRange1 = 64.0;
float Gforce[3];
float Gforce1[3];
float SMA;
float SVM;
float SMA1;
float SVM1;
int16_t ax, ay, az;
int16_t gx, gy, gz;
```

```
int16_t axx, ayy, azz;
int valx, valy, valz;
int valx1, valy1, valz1;
double accX, accY, accZ;
double gyroX, gyroY, gyroZ;
double gyroXangle, gyroYangle;
double compAngleX, compAngleY;
double kalAngleX, kalAngleY;
double CalX, CalY; // Calibarated Values
double OffsetX, OffsetY;// offset values
uint32_t timer;
double Step_size = 0.0;
uint8_t i2cData[14];
bool blinkState = false;

void kalaman();
void getG();
void getG1();
void workdone();
void calibrate();

void setup(){
    Wire.begin();
    fall = 0;
   TWBR = ((F_CPU / 400000L) - 16) / 2;

  i2cData[0] = 7;
  i2cData[1] = 0x00;
  i2cData[2] = 0x00;
```

```
i2cData[3] = 0x00;

  Serial.begin(115200);
  Serial.println("Initializing Card");
  pinMode(CS_pin, OUTPUT);
  if (!SD.begin(CS_pin))
{
    Serial.println("Card Failure");
    return;
}
Serial.println("Card Ready");

  // initialize device
  Serial.println("Initializing I2C devices...");
  accel.initialize();
  accelgyro.initialize();


   // verify connection
  pinMode(LED_PIN, OUTPUT);
  //cofe from kalaman
  delay(100); // Wait for sensor to stabilize
 File logFile = SD.open("LOG.csv", FILE_WRITE);
if (logFile)
{
  logFile.println(",,,,,,,");
  String header =
  "Time, SVM, SMA, SVM1, SMA1, Xangle,
  Yangle, Fall_Detected";
```

```
    logFile.println(header);
    logFile.close();
    Serial.println(header);
  }
  else
  {
    Serial.println("Couldn't open log file");
  }


 /* Set kalman and gyro starting angle */
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  accX = ax;
  accY = ay;
  accZ = az;

  #ifdef RESTRICT_PITCH
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch =
   atan(-accX / sqrt(accY * accY + accZ * accZ))
   * RAD_TO_DEG;
#else
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ))
   * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  kalmanX.setAngle(roll);
  kalmanY.setAngle(pitch);
```

```
 gyroXangle = roll;
 gyroYangle = pitch;
 compAngleX = roll;
 compAngleY = pitch;

 timer = micros();

 int calsel =1;
}

 void loop(){
 kalaman();
 if(calsel == 1){
   delay(1000);
 OffsetX = kalAngleX;
 OffsetY = kalAngleY;
 calsel = 0;
 }
 CalX = kalAngleX - OffsetX;
 CalY = kalAngleY - OffsetY;
 Serial.println(CalX);
 Serial.println(CalY);
 Serial.println(OffsetX);
 Serial.println(OffsetY);
 accelgyro.setFullScaleAccelRange(MPU6050_ACCEL_FS_8);
 accel.setRange( ADXL345_RANGE_8G );
 delay(10);
 accel.getAcceleration(&axx, &ayy, &azz);
 valx1 = axx;
```

```
valy1 = ayy;
valz1 = azz;
accelgyro.getAcceleration(&ax, &ay, &az);
valx = ax;
valy = ay;
valz = az;
workdone();

File logFile = SD.open("LOG.csv", FILE_WRITE);
if (logFile)
{
  logFile.print(Step_size);
  logFile.print(",");
  logFile.print(SVM);
  logFile.print(",");
  logFile.print(SMA);
  logFile.print(",");
  logFile.print(SVM1);
  logFile.print(",");
  logFile.print(SMA1);
  logFile.print(",");
  logFile.print(CalX);
  logFile.print(",");
  logFile.print(CalY);
  logFile.print(",");
  logFile.println(fall);
  logFile.close();
}
else
```

```
{
  Serial.println("Couldn't open log file");
}
    Step_size = (double)(micros() - timer)/1000000;
//Increment ID number
fall = 0;
id++;
}



   void kalaman ()
   {
     // read raw accel measurements from device
     accelgyro.setFullScaleAccelRange(0);
     delay(10);
  accel.getAcceleration(&axx, &ayy, &azz);
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  accX = ax;
  accY = ay;
  accZ = az;
  gyroX = gx;
  gyroY = gy;
  gyroZ = gz;

  double dt = (double)(micros() - timer) / 1000000;
timer = micros();
```

```
#ifdef RESTRICT_PITCH
  double roll  = atan2(accY, accZ) * RAD_TO_DEG;
  double pitch =
   atan(-accX / sqrt(accY * accY + accZ * accZ))
   * RAD_TO_DEG;
#else
  double roll  = atan(accY / sqrt(accX * accX + accZ * accZ))
   * RAD_TO_DEG;
  double pitch = atan2(-accX, accZ) * RAD_TO_DEG;
#endif

  double gyroXrate = gyroX / 131.0;
  double gyroYrate = gyroY / 131.0;
#ifdef RESTRICT_PITCH
  if ((roll < -90 && kalAngleX > 90) || (roll > 90
  && kalAngleX < -90)) {
    kalmanX.setAngle(roll);
    compAngleX = roll;
    kalAngleX = roll;
    gyroXangle = roll;
  } else
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);

  if (abs(kalAngleX) > 90)
    gyroYrate = -gyroYrate;
  kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);
#else
  if ((pitch < -90 && kalAngleY > 90) || (pitch > 90
```

```
    && kalAngleY < -90)) {
      kalmanY.setAngle(pitch);
      compAngleY = pitch;
      kalAngleY = pitch;
      gyroYangle = pitch;
    } else
      kalAngleY = kalmanY.getAngle(pitch, gyroYrate, dt);

    if (abs(kalAngleY) > 90)
      gyroXrate = -gyroXrate;
    kalAngleX = kalmanX.getAngle(roll, gyroXrate, dt);
#endif

    gyroXangle += gyroXrate * dt;
    gyroYangle += gyroYrate * dt;
    //gyroXangle += kalmanX.getRate() * dt;
    //gyroYangle += kalmanY.getRate() * dt;

    compAngleX =
    0.93 * (compAngleX + gyroXrate * dt) + 0.07 * roll;
    compAngleY =
    0.93 * (compAngleY + gyroYrate * dt) + 0.07 * pitch;


    if (gyroXangle < -180 || gyroXangle > 180)
      gyroXangle = kalAngleX;
    if (gyroYangle < -180 || gyroYangle > 180)
      gyroYangle = kalAngleY;
     Serial.print("Kalaman Angle X-axis:  ");
```

```
  Serial.print(kalAngleX);
  Serial.print("\t");

 Serial.print("\t");

Serial.print("Kalaman Angle Y-axis:  ");
Serial.print(kalAngleY);
Serial.print("\t");


 Serial.print("\r\n");
 delay(2);

   blinkState = !blinkState;
   digitalWrite(LED_PIN, blinkState);
    }

void workdone(){
 uint8_t rate;
rate = accelgyro.getFullScaleAccelRange();
Serial.println(rate);
getG();
getG1();
for(int i=0; i<3;i++){
  Serial.print("G-MPU  ");
Serial.println(Gforce[i]);
}
for(int i=0; i<3;i++){
  Serial.print("G-ADLX   ");
```

```
 Serial.println(Gforce1[i]);
 }

       SMA1 = SMA_FORM(Gforce1[0], Gforce1[1], Gforce1[2]);
       SVM1 = SVM_FORM(Gforce1[0], Gforce1[1], Gforce1[2]);
       SMA = SMA_FORM(Gforce[0], Gforce[1], Gforce[2]);
       SVM = SVM_FORM(Gforce[0], Gforce[1], Gforce[2]);


    if (SVM1 <= 0.744 && SMA1 <= 0.9197 && SVM
   <= 0.8182 && SMA <= 0.9297 ){
      Serial.println("fall detected");
      fall = 1;
      }
}

void getG(){
  Gforce[0] = valx/divRange;  Gforce[1] = valy/divRange;
  Gforce[2] = valz/divRange;
  }
void getG1(){
   Gforce1[0] = valx1/divRange1;
  Gforce1[1] = valy1/divRange1;
  Gforce1[2] = valz1/divRange1;
  }
```